IDKit SDK

Version 9.2.3.0

Table of Contents

Overview 1

Usage Overview 1

Product Differences 2

Supported Platforms 3

Similarity Scores 4

Identification Speed 4

Database Connectivity 5

Image Processing 6

C++ API Reference 7

Functions 7

Initialization 8

IEngine_InitModule Function 8

IEngine_InitWithLicense Function 8

IEngine_TerminateModule Function 9

Connections 9

IEngine_InitConnection Function 9

IEngine_SelectConnection Function 10

IEngine_Connect Function 10

IEngine_CloseConnection Function 11

User Record 11

Allocation 12

IEngine_InitUser Function 12

IEngine_CopyUser Function 12

IEngine_ClearUser Function 13

IEngine_FreeUser Function 13

Add/Remove 14

Add/Remove Fingerprint 14

Add/Remove Face 21

Add/Remove Iris 33

Properties 40

Fingerprint Properties 40

Face Properties 45

Iris Properties 46

IEngine_SetCustomData Function 48

IEngine_GetCustomData Function 49
Biometry 49
IEngine_GetFingerprintPresence Function 50
IEngine_GetFingerprintPresenceRAW Function 50
IEngine_GetFingerprintQuality Function 51
IEngine_GetDeltasAndCores Function 51
IEngine_GetMinutiaeImage Function 52
IEngine_SaveMinutiaeImage Function 53
IEngine_GetFaceQuality Function 54
IEngine_GetIrisQuality Function 54
Import/Export 55
IEngine_SerializeUser Function 55
IEngine_DeserializeUser Function 56
IEngine_ExportUserTemplate Function 56
IEngine_ImportUserTemplate Function 57
Tags 58
IEngine_SetIntTag Function 58
IEngine_GetIntTag Function 59
IEngine_SetStringTag Function 59
IEngine_GetStringTag Function 60
IEngine_HasTag Function 61
IEngine_ClearTag Function 61
IEngine_GetTagCount Function 62
IEngine_GetTagName Function 62
Collections of Users 63
IEngine_InitCollection Function 63
IEngine_FreeCollection Function 64
IEngine_GetCollectionSize Function 64
IEngine_GetCollectionIDs Function 64
Database 65
IEngine_RegisterUser Function 65
IEngine_RegisterUserAs Function 66
IEngine_UpdateUser Function 67
IEngine_UserExists Function 68
IEngine_GetUserIfExists Function 68
IEngine_GetUser Function 69
IEngine_RemoveUser Function 69
IEngine_ClearDatabase Function 70
IEngine_GetUserCount Function 70
IEngine_GetAllUserIDs Function 71
IEngine_GetUserIDsByQuery Function 71
Identification 72

IEngine_FindUser Function 72
IEngine_FindUserInSelection Function 73
IEngine_FindUserByQuery Function 74
IEngine_FindUserInMemory Function 74
IEngine_FindFingerprint Function 75
IEngine_FindFingerprintInSelection Function 76
IEngine_FindFingerprintByQuery Function 77
IEngine_FindFingerprintInMemory Function 78
Verification 79
IEngine_MatchFingerprint Function 79
IEngine_MatchFingerprints Function 80
IEngine_MatchFingerprints_transformation Function 81
IEngine_MatchFace Function 82
IEngine_MatchFaces Function 83
IEngine_MatchIris Function 84
IEngine_MatchIrises Function 85
IEngine_MatchUser Function 86
IEngine_MatchUsers Function 87
IEngine_MatchUsersModalities Function 88
Images 89
IEngine_ConvertRawToImage Function 89
IEngine_ConvertImageToRaw Function 90
IEngine_ConvertImage Function 91
IEngine_GetFaceTemplate Function 92
Settings and Constants 92
IEngine_SetParameter Function 93
IEngine_GetParameter Function 93
IEngine_SetStringParameter Function 94
IEngine_GetStringParameter Function 94
IEngine_SetCryptKey Function 95
IEngine_SetLogFile Function 95
IEngine_GetProductString Function 96
IEngine_GetHardwareId Function 96
IEngine_GetLicenseInformation Function 96
IEngine_GetErrorMsg Function 97
IEngine_GetMemoryUsage Function 97
IEngine_GetUserLimit Function 98
IEngine_GetHardwareIdByMethod Function 98
IEngine_GetFingerprintClass Function 99
IEngine_GetFingerprintResizeRatio Function 99
IEngine_GetIntermediateImages Function 100
IEngine_GetIrisTemplate Function 100

IEngine_GetMinutiaePoints Function 101
IEngine_MatchUsersModalitiesWithIntermediates Function 101
IEngine_MatchUsersWithIntermediates Function 102
IEngine_SetTracingSpanContextJson Function 102

Types, Structures, Enumerations 102
IENGINE_COLLECTION Type 103
IENGINE_CONFIG Enumeration 103
IENGINE_CONNECTION Type 107
IENGINE_FINGER_POSITION Enumeration 107
IENGINE_IMAGE_FORMAT Enumeration 108
IENGINE_MINUTIAE Structure 108
IENGINE_CRITICAL_POINT Structure 109
IENGINE_TEMPLATE_FORMAT Enumeration 109
IENGINE_USER Type 110
CRYPT_KEY_LENGTH Macro 110
MAX_CRITICAL_POINTS_COUNT Macro 110
IENGINE_FINGERPRINT_CLASS Enumeration 110
IENGINE_EXTRACTOR_ALGORITHM Enumeration 111
IENGINE_IRIS_IMAGE_TYPE Type 111
IENGINE_IRIS_POSITION Type 111
IENGINE_MATCHING_MODALITY Enumeration 112
IENGINE_HARDWARE_ID_METHOD Enumeration 112

Function Quick Reference 113

Error Codes 115

Notes 118

Buffers (C++) 118

Connections 119

Connection String 119

Corresponding Fingerprints 120

Corresponding Irises 121

Custom Data String 121

External Database 121

FAR and Threshold 122
Verification Threshold 123
Identification Threshold 124

Biometric Template 124

Fingerprint Template 126
Face Template 127
Iris Template 129

FRR **129**

Identification (one-to-many) **129**

In-memory License **130**

License **130**

Fingerprint Quality and Presence **131**

Face Quality **132**

Iris Quality **132**

Score Consolidation **132**

Selection **132**

Similarity Score **133**

Tag Query **133**

Tags **135**

Template Cache **136**

Template Extraction **136**

Threads **136**

Verification (one-to-one) **137**

Similarity Threshold **137**

Open source SW credits 139

LibSVM **139**

Caffe **139**

OpenCV **140**

GTest **140**

GFlags **141**

GLog **141**

OpenBlas **142**

Protobuf **143**

Global Matting **143**

Boost **144**

libpng 144

zlib 146

Jasper 147

NBIS 147

JPEG Group 148

SQLite3 149

Android NDK 150

JNA 152

mbed TLS 153

miniglog 155

polly 156

OXFORD VGGFace2 Dataset 156

LevelDB 157

Catch2 157

FakeIt 158

OpenSSL 158

bison 160

bzip2 160

flex 161

gsl_microsoft 162

jaeger-client-cpp 162

jsonformoderncpp 165

libevent 166

opentracing-cpp 167

optional-lite 170

thrift 171

yaml-cpp 174

xsimd 175

xercesc 175

openjpeg 178

crc32c 179

snappy **179**

libtiff **180**

icu **180**

Intel MKL **181**

Index a

1 Overview

1.1 Usage Overview

The IDKit library can be used in all types of applications requiring fingerprint, face or iris recognition.

User support

The use of the SDK is made simple through the notion of **user**. A user is a collection of biometric samples - fingerprint, face and iris images from different fingers, face and irises. This is very universal, as you can associate many fingerprint, face or iris images with a particular user. You can for example create a user with 2 fingerprints per each finger, which makes 2x10 = 20 fingerprint images in total. Through verification functions you can directly compare two users without needing to explicitly combine similarity scores from different fingers - this is done automatically by the SDK.

Database support

In addition, **database** management functions are available that enable to store and retrieve user information in/from a database. In the database, you may optionally **store both** biometric **templates** and **fingerprint images** (face and iris images can't be stores in database). Biometric templates contain biometric features necessary for verification or identification. Fingerprint images are not required for fingerprint recognition but it may be an interesting option to store them in the database and display them when the user is verified/identified. Another very convenient option is the possibility to store for each user custom application specific data, such as username, address, e-mail, etc. For enhanced security, you may optionally switch on the database encryption. In this case, all biometric templates, fingerprint images and custom data are automatically encrypted by AES (Advanced Encryption Standard) cipher when they are stored in the database.

Verification

The IDKit library compares biometric samples and calculates their similarity scores. **Verification** is also called "1 to 1 matching". Various verification strategies are supported:

- comparison of two biometric samples (verification of a probe fingerprint with a referenced fingerprint)
- comparison of multiple impressions (views) of the same biometric sample against one or multiple other impressions
- comparison of biometric samples coming from multiple fingers, faces or irises against another set of biometric samples

Identification

Sometimes, you need to find an unknown user in a database, knowing only one (or more) of its biometric samples. You may perform this through **identification** functions. Identification functions search for an unknown biometric sample or set of samples in the whole database. Identification search can be performed in different ways:

- identification of one particular biometric sample
- identification of multiple impressions (views) of the same biometric sample
- identification of biometric samples coming from multiple fingers, faces or irises

Identification was optimized for speed. Thanks to the automatic memory mapping of the database content, no additional database access time occurs when an identification search is performed. In addition, the IDKit library contains a proprietary high speed fingerprint identification algorithm, suitable for the most demanding applications.

Supported image formats

The IDKit library accepts as an input biometric samples stored in different image formats.

For fingerprint images it supports BMP, PNG, JPG, JPEG2K, GIF, TIFF and WSQ images. WSQ is a special format designed specially for efficient fingerprint image compression.

For face and iris images IDKit Multi SDK supports BMP, PNG, JPG, JPEG2K, TIFF and WSQ images.

Biometric modalities

IDKit Multi SDK is based on IDKit PC SDK and it is extended by functionality of face and iris template extraction, verification (1:1) and identification (1:N). It allows developers, system integrators and solution providers to harness powerful multimodal biometric capabilities in one flexible, user-friendly development solution.

.NET and Java connector

.NET and Java connectors with documentation and samples are provided with the SDK enabling the developers to implement applications on different platforms.

1.2 Product Differences

This documentation covers following products. Product compatibility notices appear throughout this documentation wherever there are differences between IDKit SDK versions.

IDKit PC SDK

IDKit PC SDK is **simple** to use and **straightforward**. Thanks to the **database support**, the programmer does not have to worry about storing fingerprint images or templates. Database is automatically loaded and mapped in memory, enabling **extremely fast fingerprint matching**. IDKit PC SDK is suitable for developing fingerprint **identification applications** ranging from a few up to thousands of users. IDKit PC SDK supports a wide range of **input image formats** (RAW, PNG, BMP, JPEG, JPEG 2000, GIF, WSQ, TIF) and is compatible with fingerprint images from **various fingerprint scanners** (optical, capacitive, thermal based).

IDKit Android SDK

IDKit Android SDK is port of IDKit PC SDK for Android platforms. In older versions of IDKit Android SDK only RAW and BMP image formats are supported. Since version 3.0.9 IDKit Android SDK also supports WSQ, PNG and JPEG2K image formats. IDKit Android SDK provides valid identification speed range 0-4.

IDKit Embedded SDK

IDKit Embedded SDK is port of IDKit PC SDK for embedded platforms based on embedded Linux. It is built upon customer request using specific toolchain (contains cross-compiler) which should be provided by the customer. In older versions of IDKit Embedded SDK only RAW and BMP image formats are supported. Since version 3.0.9 IDKit Embedded SDK usually (if it is supported by particular system) also supports WSQ, PNG and JPEG2K image formats. IDKit Embedded SDK provides valid identification speed range 0-4.

IDKit Multi SDK

IDKit Multi SDK is based on IDKit PC SDK and it is extended by functionality of face and iris template extraction, verification (1:1) and identification (1:N). It allows developers, system integrators and solution providers to harness powerful multimodal biometric capabilities in one flexible, user-friendly development solution.

IDKit Multi Android SDK

IDKit Multi Android SDK is based on IDKit Android SDK and it is extended by functionality of face and iris template extraction, verification (1:1) and identification (1:N). It allows developers, system integrators and solution providers to harness powerful multimodal biometric capabilities in one flexible, user-friendly development solution.

Related Products

This documentation doesn't cover several other products that can be used together with IDKit SDK. These products have their own documentation. They are briefly described below.

ANSI/ISO SDK

Innovatrics ANSI/ISO SDK can extract and match templates conforming to ANSI/INCITS 378 or ISO/IEC 19794-2 standard. IDKit SDK can export ANSI and ISO templates to be used in ANSI/ISO SDK. It is not possible to convert ANSI or ISO template back to IDKit SDK template.

Segmentation SDK

Innovatrics Segmentation SDK can perform segmentation of slap fingerprint image into individual fingerprints. These individual fingerprints can then be used in IDKit SDK for fingerprint template extraction, verification (1:1) and identification (1:N). Segmentation SDK also provides functionality for various image quality acquisition.

1.3 Supported Platforms

IDKit SDK

Innovatrics IDKit SDK is officially supported for following OS and architectures

- Windows 32-bit and 64-bit - IDKit PC SDK
  - Windows 7 and higher
- Linux 32-bit and 64-bit - IDKit PC SDK
  - Red Hat, Centos - version 7 and higher
- Android - IDKit Android SDK

Architecture	NDK API Android OS
----------	---- ---------- ----------
armv7a, x86	r23 21 and higher 5 and higher
arm64-v8a, x86_64	r23 21 and higher 5 and higher

IDKit Multi SDK

Innovatrics IDKit Multi SDK is officially supported for following OS and architectures

- Windows 64-bit
  - Windows 7 and higher
- Linux 64-bit
  - Red Hat, Centos - version 7 and higher
- Android - IDKit Multi Android SDK

Architecture	NDK API Android OS
----------	---- ---------- ----------
armv7a, x86	r23 21 and higher 5 and higher
arm64-v8a, x86_64	r23 21 and higher 5 and higher

Notes

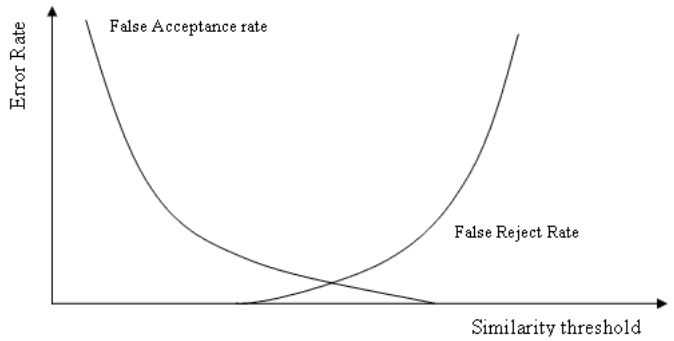
If you are interested in porting of IDKit SDK or IDKit Multi SDK for other platform please contact your sales representative or sales@innovatrics.com.

1.4 Similarity Scores

Due to different positioning, deformations, random noise, finger humidity and sensor conditions it is impossible for two images of the same finger, acquired in different sessions, to coincide exactly. For this reason, the matching is performed by an algorithm which calculates global **similarity score**. This similarity score has to be compared with the **similarity threshold**: When similarity score is greater than the similarity threshold, the system claims that the two samples coincide. Sometimes the output of biometric systems can be incorrect; the main system errors are usually measured in terms of:

- **FAR** (False Acceptance Rate (see page 122)) - The frequency of fraudulent accesses due to impostors claiming a false identity
- **FRR (see page 129)** (False Rejection Rate (see page 129)) - The frequency of rejections relative to people who should be correctly verified

FAR and FRR (see page 129) depend on the similarity threshold, which is set to obtain the desired security level. FAR and FRR (see page 129) are strictly related to each other. More precisely, FRR (see page 129) is an increasing function of the similarity threshold and FAR is a decreasing function of the similarity threshold. If the similarity threshold setting is increased to make it harder for impostors to gain access (lower FAR), some authorized people may find it harder to gain access (higher FRR (see page 129)).



The range of returned similarity scores is between 0 (no similarity) and 1000 (highest similarity). The correspondence between fingerprint templates is generally reflected by similarity scores greater than default similarity threshold (40). For more information please refer to FAR and Threshold (see page 122).

Default similarity threshold value (40) roughly corresponds to FAR=10^-4. Depending on the type of scanner, the size of fingerprint images, the size of the database and the desired security level of your application, you may need to select a different threshold. To modify the default similarity threshold, call function IEngine_SetParameter (see page 93) with parameter CFG_SIMILARITY_THRESHOLD (see page 103).

Please contact Innovatrics Technical Support for more details about setting the correct similarity threshold.

1.5 Identification Speed

Identification algorithm speed is measured in number of fingerprint comparisons per second (number of compared fingerprint pairs per second). Note that this is not the same as number of user comparisons per second, because each user pair may

contain multiple corresponding fingerprint pairs.

Performance of identification functions included in this library depends on general settings of the library - more precisely on two particular parameters set through IEngine_SetParameter (see page 93) function:

- maximum permitted rotation (see page 103) between fingerprints
- selected identification speed level (see page 103) and
- selected best identification candidates count (see page 103).

From theoretical point of view, the time complexity of the library's identification functions is given as T = a.n + b where T is the total identification time, n is the number of compared fingerprints (database size) and a, b are constant coefficients. Coefficient b represents the functions' initialization overhead and it is typically smaller than 50 milliseconds. As n grows larger, the a.n term dominates and b becomes irrelevant.

Performance may vary with population and hardware. When measuring identification speed of your system, it is recommended to use bigger databases in order to avoid the negative influence of the b coefficient.

1.6 Database Connectivity

IDKit SDK can access fingerprints from several data sources:

- integrated SQLite database stored in one file,
- memory area which is filled with fingerprint data the application needs to search.

Note

Since IDKit SDK version 8, ODBC and JDBC databases and IDispatcher service connection are not available and unsupported.

Selecting Database

Database connection is made using function IEngine_Connect (see page 10). This function takes a string parameter which describes the data source. Once connected, all IDKit SDK functions work identically with all data sources. The same API is used to access all types of databases.

Case 1: SQLite Database

In the first case, biometric data can be stored in local file '*idkit.db*' in SQLite database. IDKit SDK is initialized with this code:

```
IEngine_InitModule();
IEngine_Connect( "idkit.db" );
```

Case 2: No Database

If the application is designed to store biometric data in its own database, IDKit SDK can be instructed to use biometric data loaded by the application into memory. IDKit SDK is initialized to use memory database:

```
IEngine_InitModule();
IEngine_Connect( "Type=Memory" );
```

This initializes an empty memory database for storing biometric data. Application can then use function IEngine_RegisterUserAs (see page 66) to fill the memory database with user records obtained from application database. Any modifications of user records in memory database will be lost when application terminates. Application should save any modified records back into application database.

Another option is to avoid calls to IEngine_Connect (see page 10) entirely and use identification and verification functions

that can work with templates from external database (see page 121).

1.7 Image Processing

This section deals with various issues related to working with images.

Supported Image Formats

Supported image formats are enumerated in IENGINE_IMAGE_FORMAT (see page 108) in C++ and ImageFormat in .NET. In addition, IDKit SDK supports RAW image format through functions IEngine_ConvertRawToImage (see page 89) and IEngine_ConvertImageToRaw (see page 90) in C++ or methods IDKit.ConvertRawToImage and IDKit.ConvertImageToRaw in .NET.

RAW Image

Fingerprint RAW Image

Raw 8-bit grayscale images are canonically encoded. The minimum value that will be assigned to a "black" pixel is zero. The maximum value that will be assigned to a "white" pixel is 255. Intermediate gray levels will have assigned values of 1- 254. The pixels are stored left to right, top to bottom, with one 8-bit byte per pixel. The number of bytes in an image is equal to its height multiplied by its width as measured in pixels; there is no header. The image height and width in pixels will be supplied to the SDK as supplemental information.

IDKit SDK contains functions for converting from standard uncompressed BMP image format to the raw format described above.

Face and Iris RAW Image

Raw 24bit (8bit per kanal) per pixel images are supported for face and iris modality in IDKit Multi SDK.

BMP Support

BMP image format can be found in multiple variants. In IDKit SDK following BMP variants are supported:

- 8bit - palette mapped to BGR,
- 24bit - standard uncompressed BGR.

During the BMP image extraction only the green channel of image is processed.

Image Conversions

Images can be supplied to IDKit SDK and requested from IDKit SDK in any supported format. IDKit SDK performs image format conversion whenever necessary.

Image conversion is sometimes enforced by IDKit SDK. IDKit SDK and ExpressID AFIS always store fingerprint images in database in PNG format. If different image format is supplied, it is converted to PNG format upon write to database. If PNG format is not supported on some platform, BMP is used as the default format for storing fingerprint images in database (for example in IDKit Embedded SDK).

2 C++ API Reference

Functions

	Name	Description
	Initialization (see page 8)	
	Connections (see page 9)	
	User Record (see page 11)	
	Collections of Users (see page 63)	
	Database (see page 65)	
	Identification (see page 72)	
	Verification (see page 79)	
	Images (see page 89)	
	IEngine_GetFaceTemplate (see page 92)	Exports face template at specified index from user structure.
	Settings and Constants (see page 92)	
	IEngine_GetFingerprintClass (see page 99)	Analyzes fingerprint image and returns its class according to IAFIS specification (https://identifyus.org/help/FPinformation.pdf).
	IEngine_GetFingerprintResizeRatio (see page 99)	Unsupported function for internal purposes. Fingerprint image can be rescaled prior to extraction in the case when ridge frequency is too low and if CFG_EXTRACT_MULTISCALE is set to 1. This function returns resize factor (if any) calculated based on the ridge distance analysis. This feature is used for better accuracy with children's fingerprints.
	IEngine_GetIntermediateImages (see page 100)	Unsupported function for internal purposes
	IEngine_GetIrisTemplate (see page 100)	Exports iris template at specified index from user structure.
	IEngine_GetMinutiaePoints (see page 101)	Returns list of fingerpint minutiae points (endings and bifurcations)
	IEngine_MatchUsersModalitiesWithIntermediates (see page 101)	Unsupported function for internal purposes
	IEngine_MatchUsersWithIntermediates (see page 102)	Unsupported function for internal purposes
	IEngine_SetTracingSpanContextJson (see page 102)	Unsupported function for internal purposes

2.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	IEngine_GetFaceTemplate (see page 92)	Exports face template at specified index from user structure.
	IEngine_GetFingerprintClass (see page 99)	Analyzes fingerprint image and returns its class according to IAFIS specification (https://identifyus.org/help/FPinformation.pdf).
	IEngine_GetFingerprintResizeRatio (see page 99)	Unsupported function for internal purposes. Fingerprint image can be rescaled prior to extraction in the case when ridge frequency is too low and if CFG_EXTRACT_MULTISCALE is set to 1. This function returns resize factor (if any) calculated based on the ridge distance analysis. This feature is used for better accuracy with children's fingerprints.
	IEngine_GetIntermediateImages (see page 100)	Unsupported function for internal purposes
	IEngine_GetIrisTemplate (see page 100)	Exports iris template at specified index from user structure.
	IEngine_GetMinutiaePoints (see page 101)	Returns list of fingerpint minutiae points (endings and bifurcations)
	IEngine_MatchUsersModalitiesWithIntermediates (see page 101)	Unsupported function for internal purposes
	IEngine_MatchUsersWithIntermediates (see page 102)	Unsupported function for internal purposes
	IEngine_SetTracingSpanContextJson (see page 102)	Unsupported function for internal purposes

2.1.1 Initialization

Functions

	Name	Description
	IEngine_InitModule (see page 8)	Initializes the library
	IEngine_InitWithLicense (see page 8)	Initializes the library with selected (in-memory (see page 130)) license
	IEngine_TerminateModule (see page 9)	Terminates the use of the library

2.1.1.1 IEngine_InitModule Function

Initializes the library

C++

```
IDKIT_API int IEngine_InitModule();
```

Description

This function initializes and checks the integrity of the library and verifies the validity of the license (see page 130). It should be called prior to any other function from the library except IEngine_GetHardwareId (see page 96) and IEngine_GetProductString (see page 96).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADLICENSE	Provided license is not valid, or no license was found.
IENGINE_E_EXPIREDLICENSE	Provided license has expired.
IENGINE_E_MISSINGDLL	At least one required DLL could not be loaded.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
Other	ILicense error code.

Related Topics

IEngine_TerminateModule (see page 9)

2.1.1.2 IEngine_InitWithLicense Function

Initializes the library with selected (in-memory (see page 130)) license

C++

```
IDKIT_API int IEngine_InitWithLicense(const unsigned char * license, int length);
```

Parameters

`const unsigned char * license`

[in] Buffer containing an in-memory license (see page 130).

`int length`

[in] Length of the buffer containing license data.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADLICENSE	Provided license is not valid.
IENGINE_E_EXPIREDLICENSE	Provided license has expired.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADVALUE	Invalid value provided.

`Other`	ILicense error code.

Remarks

This function initializes and checks the integrity of the library and verifies validity of the provided in-memory license (see page 130). It should be called prior to any other function from the library except IEngine_GetHardwareId (see page 96) and IEngine_GetProductString (see page 96).

Related Topics

IEngine_TerminateModule (see page 9), IEngine_InitModule (see page 8)

2.1.1.3 IEngine_TerminateModule Function

Terminates the use of the library

C++

```
IDKIT_API int IEngine_TerminateModule();
```

Description

This function releases all resources allocated by the library. It should be called as the very last function of the library.

Return Values

Return Values	Description
`IENGINE_E_NOERROR`	No error occurred.
`IENGINE_E_INTERNAL`	Unspecified internal error occurred.

Related Topics

IEngine_InitModule (see page 8)

2.1.2 Connections

Functions

	Name	Description
	IEngine_InitConnection (see page 9)	Initializes new connection to database.
	IEngine_SelectConnection (see page 10)	Selects active connection for the current thread.
	IEngine_Connect (see page 10)	Connects to a database containing biometric data.
	IEngine_CloseConnection (see page 11)	Closes a connection and releases all resources held by the connection.

2.1.2.1 IEngine_InitConnection Function

Initializes new connection to database.

C++

```
IDKIT_API IENGINE_CONNECTION IEngine_InitConnection();
```

Returns

Handle to the newly created connection or NULL on error.

Return Values

Return Values	Description
`NULL`	Memory error - could not allocate memory.
`IENGINE_CONNECTION (see page 107)`	An initialized IENGINE_CONNECTION (see page 107) structure in disconnected state.

Remarks

Applications that need only one connection never need to call this function, because the default connection is created and selected automatically. Applications that need multiple connections (see page 119) can call this function multiple times to create as many connections as they need.

The new connection is not initially selected for any thread (see page 136). Use IEngine_SelectConnection (see page 10) to select it for the current thread. The connection is initially in disconnected state. Call IEngine_Connect (see page 10) to establish connection to a database.

Connections may be left in disconnected state in which case they are useful as a container for connection-specific parameters. Every thread can then have a different set of global parameters.

Related Topics

IEngine_SelectConnection (see page 10), IEngine_Connect (see page 10), IEngine_CloseConnection (see page 11)

Availability

Supported in IDKit PC SDK and IDKit Multi SDK.

2.1.2.2 IEngine_SelectConnection Function

Selects active connection for the current thread.

C++

```
IDKIT_API int IEngine_SelectConnection(const IENGINE_CONNECTION connection);
```

Parameters

`const IENGINE_CONNECTION connection`

[in] Connection to be selected or NULL to select default connection.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Remarks

Every thread (see page 136) has its selected connection (see page 119). All API functions operate on the currently selected connection. Only one connection can be selected by a given thread, but multiple threads can share the same connection. By default, all threads have selected default connection. You can call this function with connection handle returned from IEngine_InitConnection (see page 9) to select non-default connection or with NULL parameter to select default connection again.

Related Topics

IEngine_InitConnection (see page 9), IEngine_Connect (see page 10), IEngine_CloseConnection (see page 11)

Availability

Supported in IDKit PC SDK and IDKit Multi SDK.

2.1.2.3 IEngine_Connect Function

Connects to a database containing biometric data.

C++

```
IDKIT_API int IEngine_Connect(const char * connectionString);
```

Description

This function connects IDKit SDK to the database specified in connection string (see page 119). For local databases, it also fills template cache (see page 136).

Parameters

`const char * connectionString`

[in] Specifies the connection string (see page 119) used to connect to database.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NULLPARAM	Parameter connectionString must not be NULL.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_DBFULL	Exceeded database user limit.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_BADCRYPTKEY	Invalid cipher key. Data CRC check failed after decryption.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
Other	ILicense error code.

Related Topics

IEngine_ClearDatabase (see page 70), IEngine_InitConnection (see page 9)

Availability

Connections to JDBC/ODBC database and ExpressID AFIS are not available and unsupported since IDKit SDK version 8.

2.1.2.4 IEngine_CloseConnection Function

Closes a connection and releases all resources held by the connection.

C++

```
IDKIT_API int IEngine_CloseConnection(IENGINE_CONNECTION connection);
```

Parameters

`IENGINE_CONNECTION connection`

[in] Connection to be closed.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Remarks

This function should be called when connection (see page 119) created by IEngine_InitConnection (see page 9) is no longer needed. The connection doesn't have to be selected by the current thread. If any API call is executed while the destroyed connection is still selected, the API function returns error.

Related Topics

IEngine_InitConnection (see page 9), IEngine_SelectConnection (see page 10)

Availability

Supported in IDKit PC SDK and IDKit Multi SDK.

2.1.3 User Record

2.1.3.1 Allocation

Functions

	Name	Description
	IEngine_InitUser (see page 12)	Initializes an empty user structure
	IEngine_CopyUser (see page 12)	Copies user structure
	IEngine_ClearUser (see page 13)	Clears all information from user structure
	IEngine_FreeUser (see page 13)	Releases user structure

2.1.3.1.1 IEngine_InitUser Function

Initializes an empty user structure

C++

```
IDKIT_API IENGINE_USER IEngine_InitUser();
```

Description

This function initializes an empty user structure and returns an initialized IENGINE_USER (see page 110) type. Afterwards, you may use the returned user structure and associate fingerprints (IEngine_AddFingerprint (see page 14)), faces (IEngine_AddFace (see page 22)) or irises (IEngine_AddIris (see page 34)) to this structure .

Warning

After use, don't forget to free the returned user structure by calling IEngine_FreeUser (see page 13).

Return Values

Return Values	Description
`NULL`	Memory error - could not allocate memory.
`IENGINE_USER (see page 110)`	An initialized empty IENGINE_USER (see page 110) structure.

Example

```
...
IENGINE_USER user=IEngine_InitUser();
IEngine_AddFingerprintFromFile(user,0,"image.bmp");
err=IEngine_RegisterUser(user, &userID);
if(err!=0)
        printf("Error occurred during registration\n");
else
        printf("User was registered under ID %d\n",userID);
IEngine_FreeUser(user);
```

Related Topics

IEngine_FreeUser (see page 13), IEngine_ClearUser (see page 13), IEngine_AddFingerprint (see page 14), IEngine_AddFace (see page 22), IEngine_AddIris (see page 34)

2.1.3.1.2 IEngine_CopyUser Function

Copies user structure

C++

```
IDKIT_API int IEngine_CopyUser(const IENGINE_USER srcUser, IENGINE_USER dstUser, bool
withImages);
```

Description

This function copies source user structure (biometric templates, images, custom data and tags) into destination user structure. Destination user structure is not cleared before copy, so you can use this method even for user structures joining.

Parameters

`const IENGINE_USER srcUser`
[in] source user structure

`IENGINE_USER dstUser`
[in] destination user structure

`withImage`
[in] boolean indicating whether the images are subject of copying too.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	One of user parameters is not valid.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_InitUser (see page 12), IEngine_FreeUser (see page 13), IEngine_ClearUser (see page 13), IEngine_AddFingerprint (see page 14), IEngine_AddFace (see page 22), IEngine_AddIris (see page 34)

2.1.3.1.3 IEngine_ClearUser Function

Clears all information from user structure

C++

```
IDKIT_API int IEngine_ClearUser(IENGINE_USER user);
```

Description

This function deletes all information from input user structure. All biometric templates and images are deleted from the input user structure, custom data of the user is also cleared. After calling this function, returned user structure will be an initialized empty IENGINE_USER (see page 110) structure.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_InitUser (see page 12), IEngine_FreeUser (see page 13)

2.1.3.1.4 IEngine_FreeUser Function

Releases user structure

C++

```
IDKIT_API int IEngine_FreeUser(IENGINE_USER user);
```

Description

This function releases input user structure. All data from the input user structure will be released.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

Return Values

Return Values	Description
`IENGINE_E_NOERROR`	No error occurred.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_BADUSER`	User parameter is not valid.
`IENGINE_E_INTERNAL`	Unspecified internal error occurred.

Related Topics

IEngine_InitUser (see page 12)

2.1.3.2 Add/Remove

2.1.3.2.1 Add/Remove Fingerprint

Functions

	Name	Description
	IEngine_AddFingerprint (see page 14)	Adds fingerprint to user structure, fingerprint image is read from memory
	IEngine_AddFingerprintRAW (see page 15)	Adds fingerprint to user structure, fingerprint raw image is read from memory
	IEngine_AddFingerprintFromUser (see page 16)	Adds fingerprint to user structure from another user structure.
	IEngine_AddFingerprintFromFile (see page 17)	Adds fingerprint to user structure, fingerprint image is read from file
	IEngine_SetFingerprint (see page 18)	Replaces fingerprint at specified index, fingerprint image is read from memory
	IEngine_SetFingerprintRAW (see page 18)	Replaces fingerprint at specified index, fingerprint raw image is read from memory
	IEngine_SetFingerprintFromUser (see page 19)	Replaces fingerprint at specified index with fingerprint taken from another user structure.
	IEngine_SetFingerprintFromFile (see page 20)	Replaces fingerprint at specified index, fingerprint image is read from file
	IEngine_RemoveFingerprint (see page 21)	Removes fingerprint at specified index

2.1.3.2.1.1 IEngine_AddFingerprint Function

Adds fingerprint to user structure, fingerprint image is read from memory

C++

```
IDKIT_API int IEngine_AddFingerprint(IENGINE_USER user, IENGINE_FINGER_POSITION
fingerPosition, const unsigned char * fingerprintImage, int imgSize);
```

Description

This function adds an additional fingerprint to an initialized user structure. You may add as many fingerprints as needed to one user. You may even associate more fingerprint images issued from the same finger to the same user. This helps to improve the recognition accuracy. Even if some of the fingerprints are not recognized due to high noise or deformation, other samples of the same fingerprint may still be found. This feature is also useful for multi-finger matching/identification. If you use two fingers for identification instead of one finger,the error rate will be lower.

Input fingerprint image is directly read from memory.

Each fingerprint contained in user structure has its own index. Index numbers are given automatically in ascending order, beginning at 0 (index of the first added fingerprint is 0).

Note: If your fingerprint image is stored in raw format, please use IEngine_AddFingerprintRAW (see page 15) or IEngine_ConvertRawToImage (see page 89) before calling this function

When you call this function,the fingerprint image is processed and the fingerprint template is extracted from the original image.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`IENGINE_FINGER_POSITION fingerPosition`

[in] Parameter specifying the finger position of the current fingerprint

`const unsigned char * fingerprintImage`

[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)). Minimal allowed size of image is 90x90 pixels and maximal size of image is 4000x4000 pixels.

`int imgSize`

[in] size of the fingerprint image in bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable fingerprint.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	Parameter fingerprintImage must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFingerprintRAW (see page 15), IEngine_AddFingerprintFromFile (see page 17), IEngine_AddFingerprintFromUser (see page 16), IEngine_RemoveFingerprint (see page 21), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

2.1.3.2.1.2 IEngine_AddFingerprintRAW Function

Adds fingerprint to user structure, fingerprint raw image is read from memory

C++

```
IDKIT_API int IEngine_AddFingerprintRAW(IENGINE_USER user, IENGINE_FINGER_POSITION
position, const unsigned char * rawImage, int width, int height);
```

Description

This function adds an additional fingerprint to an initialized user structure. You may add as many fingerprints as needed to one user. You may even associate more fingerprint raw images issued from the same finger to the same user. This helps to improve the recognition accuracy. Even if some of the fingerprints are not recognized due to high noise or deformation, other samples of the same fingerprint may still be found. This feature is also useful for multi-finger matching/identification. If you use two fingers for identification instead of one finger,the error rate will be lower.

Input raw fingerprint image is directly read from memory.

Each fingerprint contained in user structure has its own index. Index numbers are given automatically in ascending order, beginning at 0 (index of the first added fingerprint is 0).

When you call this function,the fingerprint image is processed and the fingerprint template is extracted from the original raw image.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 8-bit (one byte per pixel) grayscale format. The beginning is the upper left corner of the image. Value of black is 0x0, white is 0xFF. Minimal allowed size of image is 90x90 pixels and maximal size of image is 4000x4000 pixel

`int width`
[in] The number of pixels indicating the width of the raw image

`int height`
[in] The number of pixels indicating the height of the raw image

`fingerPosition`
[in] Parameter specifying the finger position of the current fingerprint

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable fingerprint.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_AddFingerprintFromFile (see page 17), IEngine_AddFingerprintFromUser (see page 16), IEngine_RemoveFingerprint (see page 21), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

2.1.3.2.1.3 IEngine_AddFingerprintFromUser Function

Adds fingerprint to user structure from another user structure.

C++

```
IDKIT_API int IEngine_AddFingerprintFromUser(IENGINE_USER user, IENGINE_USER fromUser, int fromIndex, bool withImage);
```

Description

This function adds an additional fingerprint to an initialized user structure. You may add as many fingerprints as needed to one user. This helps to improve the recognition accuracy. Even if some of the fingerprints are not recognized due to high noise or deformation, other samples of the same fingerprint may still be found. This feature is also useful for multi-finger matching/identification. If you use two fingers for identification instead of one finger,the error rate will be lower.

Each fingerprint contained in the user structure has its own index. Index numbers are given automatically in ascending order, beginning at 0 (index of the first added fingerprint is 0).

When you call this function, one fingerprint template, its position and eventually corresponding fingerprint image is get from source user structure and added into destination user structure.

Parameters

`IENGINE_USER user`
[in] An initialized destination user structure

`IENGINE_USER fromUser`
[in] An initialized source user structure

`int fromIndex`
[in] Index of the fingerprint in 'fromUser' user structure that will be added into 'user' user structure.

`bool withImage`
[in] Boolean indicating whether the fingerprint image is subject of addition too.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.

IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADTEMPLATE	Invalid user template.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_AddFingerprintRAW (see page 15), IEngine_AddFingerprintFromFile (see page 17), IEngine_RemoveFingerprint (see page 21), IEngine_InitUser (see page 12)

2.1.3.2.1.4 IEngine_AddFingerprintFromFile Function

Adds fingerprint to user structure, fingerprint image is read from file

C++

```
IDKIT_API int IEngine_AddFingerprintFromFile(IENGINE_USER user, IENGINE_FINGER_POSITION fingerPosition, const char * filename);
```

Description

This function adds an additional fingerprint to an initialized user structure. You may add as many fingerprints as needed to one user. You may even associate more fingerprint images issued from the same finger to the same user. This helps to improve the recognition accuracy. Even if some of the fingerprints are not recognized due to high noise or deformation, other samples of the same fingerprint may still be found. This feature is also useful for multi-finger matching/identification. If you use two fingers for identification instead of one finger,the error rate will be lower.

Input fingerprint image is read from the file specified on input.

Each fingerprint contained in the user structure has its own index. Index numbers are given automatically in ascending order, beginning at 0 (index of the first added fingerprint is 0).

When you call this function,the fingerprint image is processed and the fingerprint template is extracted from the original image.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`IENGINE_FINGER_POSITION fingerPosition`

[in] Parameter specifying finger position of the current fingerprint

`const char * filename`

[in] Name of the file containing fingerprint image. Image format must be one of supported image formats (Image Processing (see page 6)). Minimal allowed size of image is 90x90 pixels and maximal size of image is 4000x4000 pixel

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains a non-recognizable fingerprint.
IENGINE_E_FILE	Error occurred while opening/reading file.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	Parameter filename must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_AddFingerprintRAW (see page 15), IEngine_AddFingerprintFromUser (see page 16), IEngine_RemoveFingerprint (see page 21), IEngine_InitUser (see page 12)

2.1.3.2.1.5 IEngine_SetFingerprint Function

Replaces fingerprint at specified index, fingerprint image is read from memory

C++

```
IDKIT_API int IEngine_SetFingerprint(IENGINE_USER user, int fingerprintIndex,
IENGINE_FINGER_POSITION fingerPosition, const unsigned char * fingerprintImage, int
imgSize);
```

Description

This function replaces one of the fingerprints in an existing user structure by a new fingerprint. Previously attached fingerprint will be discarded.

Input fingerprint image is read from memory.

Fingerprint that will be replaced is defined by the fingerprintIndex parameter. Each fingerprint contained in the user structure has its own index. Index numbers are given automatically when fingerprints are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

When you call this function, fingerprint image is processed and fingerprint template is extracted from the original image.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of fingerprint that will be replaced

`IENGINE_FINGER_POSITION fingerPosition`
[in] Parameter specifying the finger position of the new fingerprint

`const unsigned char * fingerprintImage`
[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`int imgSize`
[in] Size of the fingerprint image in bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains a non-recognizable fingerprint.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	Parameter fingerprintImage must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADVALUE	Invalid fingerprintPosition value provided.
IENGINE_E_BADTEMPLATE	Invalid user template.

Related Topics

IEngine_SetFingerprintRAW (see page 18), IEngine_SetFingerprintFromFile (see page 20), IEngine_SetFingerprintFromUser (see page 19), IEngine_AddFingerprint (see page 14)

2.1.3.2.1.6 IEngine_SetFingerprintRAW Function

Replaces fingerprint at specified index, fingerprint raw image is read from memory

C++

```
IDKIT_API int IEngine_SetFingerprintRAW(IENGINE_USER user, int fingerprintIndex,
IENGINE_FINGER_POSITION fingerPosition, const unsigned char * rawImage, int width, int
height);
```

Description

This function replaces one of the fingerprints in an existing user structure by a new fingerprint. Previously attached fingerprint will be discarded.

Input fingerprint raw image is read from memory.

Fingerprint that will be replaced is defined by the fingerprintIndex parameter. Each fingerprint contained in the user structure has its own index. Index numbers are given automatically when fingerprints are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

When you call this function, fingerprint raw image is processed and fingerprint template is extracted from the original image.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int fingerprintIndex`

[in] Index of fingerprint that will be replaced

`IENGINE_FINGER_POSITION fingerPosition`

[in] Parameter specifying the finger position of the new fingerprint

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 8-bit (one byte per pixel) grayscale format. The beginning is the upper left corner of the image. Value of black is 0x0, white is 0xFF.

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains a non-recognizable fingerprint.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.

Related Topics

IEngine_SetFingerprint (see page 18), IEngine_SetFingerprintFromFile (see page 20), IEngine_SetFingerprintFromUser (see page 19), IEngine_AddFingerprint (see page 14)

2.1.3.2.1.7 IEngine_SetFingerprintFromUser Function

Replaces fingerprint at specified index with fingerprint taken from another user structure.

C++

```
IDKIT_API int IEngine_SetFingerprintFromUser(IENGINE_USER user, int index, const
IENGINE_USER fromUser, int fromIndex, bool withImage);
```

Description

This function replaces one of the fingerprints in an existing user structure by a fingerprint taken from another user structure. Previously attached fingerprint will be discarded.

Fingerprint that will be replaced is defined by the index parameter. Each fingerprint contained in user structure has its own index. Index numbers are given automatically when fingerprints are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

When you call this function, one fingerprint template, its position and eventually corresponding fingerprint image is get from source user structure and inserted into destination user structure (original fingerprint in destination user is discarded).

Parameters

`IENGINE_USER user`
[in] An initialized destination user structure

`int index`
[in] Index of fingerprint in destination user structure that will be replaced

`const IENGINE_USER fromUser`
[in] An initialized source user structure

`int fromIndex`
[in] Index of the fingerprint in 'fromUser' user structure that will be inserted into 'user' user structure

`bool withImage`
[in] Boolean indicating whether the fingerprint image is subject of insertion too.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADTEMPLATE	Invalid user template.

Related Topics

IEngine_SetFingerprintFromFile (see page 20), IEngine_AddFingerprint (see page 14), IEngine_ConvertRawToImage (see page 89)

2.1.3.2.1.8 IEngine_SetFingerprintFromFile Function

Replaces fingerprint at specified index, fingerprint image is read from file

C++

```
IDKIT_API int IEngine_SetFingerprintFromFile(IENGINE_USER user, int fingerprintIndex,
IENGINE_FINGER_POSITION fingerPosition, const char * filename);
```

Description

This function replaces one of the fingerprints in an existing user structure by a new fingerprint. Previously attached fingerprint will be discarded.

Fingerprint that will be replaced is defined by the fingerprintIndex parameter. Each fingerprint contained in user structure has its own index. Index numbers are given automatically when fingerprints are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

Note: If your fingerprint image is stored in raw format, please use IEngine_ConvertRawToImage (see page 89) before calling this function

When you call this function, fingerprint image is processed and fingerprint template is extracted from the original image.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of fingerprint that will be replaced

`IENGINE_FINGER_POSITION fingerPosition`
[in] Parameter specifying finger position of the new fingerprint

`const char * filename`
[in] Name of the file containing fingerprint image. Image format must be one of supported image formats Image Processing (see page 6).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_FILE	Error occurred while opening/reading file.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains a non-recognizable fingerprint.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter filename must not be NULL.

Related Topics

IEngine_SetFingerprint (see page 18), IEngine_SetFingerprintRAW (see page 18), IEngine_SetFingerprintFromUser (see page 19), IEngine_AddFingerprint (see page 14), IEngine_ConvertRawToImage (see page 89)

2.1.3.2.1.9 IEngine_RemoveFingerprint Function

Removes fingerprint at specified index

C++

```
IDKIT_API int IEngine_RemoveFingerprint(IENGINE_USER user, int fingerprintIndex);
```

Description

This function removes one of the fingerprints contained in input user structure.

Fingerprint that will be removed is defined by fingerprintIndex parameter. Each fingerprint contained in the user structure has its own index. Removing of a fingerprint updates all fingerprint indices which are greater than the index of removed fingerprint in such way, that indices are maintained in continuously ascending order. For instance, if a user structure has three fingerprints with respective indices 0, 1, 2 and a fingerprint with index 1 is removed, the user structure will contain two fingerprints with respective indices 0, 1. Index of a fingerprint with original value 2 was changed to have a value 1. Index numbers are given automatically when fingerprints are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int fingerprintIndex`

[in] Index of the fingerprint that will be removed

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFingerprint (see page 14)

2.1.3.2.2 Add/Remove Face

Functions

	Name	Description
	IEngine_AddFace (see page 22)	Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from memory buffer. Expected face size is absolute distance in pixels.
	IEngine_AddFaceRelative (see page 23)	Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from memory buffer. Expected face size is relative to the image frame size.
	IEngine_AddFaceRAW (see page 24)	Performs face detection and extraction of face template from face RAW image and adds it to user structure. Face RAW image is read from memory buffer. Expected face size is absolute distance in pixels.
	IEngine_AddFaceRAWRelative (see page 25)	Performs face detection and extraction of face template from face RAW image and adds it to user structure. Face RAW image is read from memory buffer. Expected face size is relative to the image frame size.
	IEngine_AddFaceFromFile (see page 26)	Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from file. Expected face size is absolute distance in pixels.
	IEngine_AddFaceFromFileRelative (see page 27)	Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from file. Expected face size is relative to the image frame size.
	IEngine_SetFace (see page 27)	Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is absolute distance in pixels.
	IEngine_SetFaceRelative (see page 28)	Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is relative to the image frame size.
	IEngine_SetFaceRAW (see page 29)	Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is absolute distance in pixels.
	IEngine_SetFaceRAWRelative (see page 30)	Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is relative to the image frame size.
	IEngine_AddFaceTemplate (see page 31)	Adds existing face template to user structure. Face template is read from memory buffer.
	IEngine_SetFaceTemplate (see page 32)	Replaces existing face template at specified index. Face template is read from memory buffer.
	IEngine_RemoveFace (see page 33)	Removes face template at specified index

2.1.3.2.2.1 IEngine_AddFace Function

Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from memory buffer. Expected face size is absolute distance in pixels.

C++

```
IDKIT_API int IEngine_AddFace(IENGINE_USER user, const unsigned char * faceImage, int
imgSize, int minFaceSize, int maxFaceSize);
```

Description

When a face is detected in input image this function extracts and adds an additional face template to an initialized user structure. You may add multiple faces to one user. You may even associate more face images issued from the same face to the same user. This helps to improve the recognition accuracy. Even if some of the face templates are not recognized due to high noise or deformation, other samples of the same face may still be found. This feature is also useful for multi-face matching/identification. If you use two face templates in one user for identification instead of one face template, the error rate will be lower.

Input face image is directly read from memory.

Only faces with specific range of sizes are detected. The face size is defined as a maximum of inter pupil distance and distance between center of mouth and center point between eyes. The range is defined by minFaceSize and maxFaceSize parameters.

Each face template contained in user structure has its own index. Index numbers are given automatically in ascending order, beginning at 0 (index of the first added face is 0).

Note: If your input image is stored in RAW format (3 bytes per pixel in BGR), please use IEngine_AddFaceRAW (see page 24). Don't use IEngine_ConvertRawToImage (see page 89), because it supports 1 byte per pixel grayscale RAW

images (fingerprint images) only.

When you call this function, the input image is processed and the face template is extracted from the original image.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`const unsigned char * faceImage`
[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`int imgSize`
[in] size of the image in bytes.

`int minFaceSize`
[in] defines minimal expected face size in input image in pixels. If set to zero, default value is used (40). Minimum non-zero valid value is 12.

`int maxFaceSize`
[in] defines maximal expected face size in input image in pixels. If set to zero, default value is used (200).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter faceImage must not be NULL.

Related Topics

IEngine_AddFaceRAW (see page 24), IEngine_AddFaceFromFile (see page 26), IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.2 IEngine_AddFaceRelative Function

Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from memory buffer. Expected face size is relative to the image frame size.

C++

```
IDKIT_API int IEngine_AddFaceRelative(IENGINE_USER user, const unsigned char * faceImage,
int imgSize, float minFaceSizeRatio, float maxFaceSizeRatio);
```

Description

When a face is detected in input image this function extracts and adds an additional face template to an initialized user structure. You may add multiple faces to one user. You may even associate more face images issued from the same face to the same user. This helps to improve the recognition accuracy. Even if some of the face templates are not recognized due to high noise or deformation, other samples of the same face may still be found. This feature is also useful for multi-face matching/identification. If you use two face templates in one user for identification instead of one face template, the error rate will be lower.

Input face image is directly read from memory.

Only faces with specific range of sizes are detected. The face size is defined as a maximum of inter pupil distance and distance between center of mouth and center point between eyes. The range is defined by minFaceSizeRatio and maxFaceSizeRatio parameters.

Each face template contained in user structure has its own index. Index numbers are given automatically in ascending order,

beginning at 0 (index of the first added face is 0).

Note: If your input image is stored in RAW format (3 bytes per pixel in BGR), please use IEngine_AddFaceRAW (see page 24). Don't use IEngine_ConvertRawToImage (see page 89), because it supports 1 byte per pixel grayscale RAW images (fingerprint images) only.

When you call this function, the input image is processed and the face template is extracted from the original image.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const unsigned char * faceImage`

[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`int imgSize`

[in] size of the image in bytes.

`float minFaceSizeRatio`

[in] defines minimal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

`float maxFaceSizeRatio`

[in] defines maximal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter faceImage must not be NULL.

Related Topics

IEngine_AddFaceRAW (see page 24), IEngine_AddFaceFromFile (see page 26), IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.3 IEngine_AddFaceRAW Function

Performs face detection and extraction of face template from face RAW image and adds it to user structure. Face RAW image is read from memory buffer. Expected face size is absolute distance in pixels.

C++

```
IDKIT_API int IEngine_AddFaceRAW(IENGINE_USER user, const unsigned char * rawImage, int
width, int height, int minFaceSize, int maxFaceSize);
```

Description

Please see documentation of IEngine_AddFace (see page 22) for more details.

Note: RAW image is RAW color 3 channels image. The rawImage is array of bytes where each pixels is formed by 3 bytes, which means 3 color channels ordered in BGR order. Origin of image is in upper-left corner. Size of the image data array can be calculated as image height *image width * 3 bytes.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 24-bit (3 bytes per pixel) BGR format. The beginning is the upper left corner of the image. Value of black is (0x00,0x00,0x00), white is (0xFF,0xFF,0xFF).

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

`int minFaceSize`

[in] defines minimal expected face size in input image in pixels. If set to zero, default value is used (40). Minimum non-zero valid value is 12.

`int maxFaceSize`

[in] defines maximal expected face size in input image in pixels. If set to zero, default value is used (200).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW, IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.4 IEngine_AddFaceRAWRelative Function

Performs face detection and extraction of face template from face RAW image and adds it to user structure. Face RAW image is read from memory buffer. Expected face size is relative to the image frame size.

C++

```
IDKIT_API int IEngine_AddFaceRAWRelative(IENGINE_USER user, const unsigned char * rawImage,
int width, int height, float minFaceSizeRatio, float maxFaceSizeRatio);
```

Description

Please see documentation of IEngine_AddFaceRelative (see page 23) for more details.

Note: RAW image is RAW color 3 channels image. The rawImage is array of bytes where each pixels is formed by 3 bytes, which means 3 color channels ordered in BGR order. Origin of image is in upper-left corner. Size of the image data array can be calculated as image height *image width * 3 bytes.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 24-bit (3 bytes per pixel) BGR format. The beginning is the upper left corner of the image. Value of black is (0x00,0x00,0x00), white is (0xFF,0xFF,0xFF).

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

`float minFaceSizeRatio`

[in] defines minimal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum

is 1.

`float maxFaceSizeRatio`

[in] defines maximal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW (see page 24), IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.5 IEngine_AddFaceFromFile Function

Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from file. Expected face size is absolute distance in pixels.

C++

```
IDKIT_API int IEngine_AddFaceFromFile(IENGINE_USER user, const char * filename, int
minFaceSize, int maxFaceSize);
```

Description

Please see documentation of IEngine_AddFace (see page 22) for more details.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const char * filename`

[in] Name of the file containing face image. Image format must be one of supported image formats (Image Processing (see page 6)).

`int minFaceSize`

[in] defines minimal expected face size in input image in pixels. If set to zero, default value is used (40). Minimum non-zero valid value is 12.

`int maxFaceSize`

[in] defines maximal expected face size in input image in pixels. If set to zero, default value is used (200).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter filename must not be NULL.
IENGINE_E_FILE	Error occurred while opening/reading file.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW (see page 24), IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.6 IEngine_AddFaceFromFileRelative Function

Performs face detection and extraction of face template from face image and adds it to user structure. Face image is read from file. Expected face size is relative to the image frame size.

C++

```
IDKIT_API int IEngine_AddFaceFromFileRelative(IENGINE_USER user, const char * filename,
float minFaceSizeRatio, float maxFaceSizeRatio);
```

Description

Please see documentation of IEngine_AddFaceRelative (see page 23) for more details.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const char * filename`

[in] Name of the file containing face image. Image format must be one of supported image formats (Image Processing (see page 6)).

`float minFaceSizeRatio`

[in] defines minimal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

`float maxFaceSizeRatio`

[in] defines maximal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter filename must not be NULL.
IENGINE_E_FILE	Error occurred while opening/reading file.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW (see page 24), IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.7 IEngine_SetFace Function

Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is absolute distance in pixels.

C++

```
IDKIT_API int IEngine_SetFace(IENGINE_USER user, int faceIndex, const unsigned char *
```

```
faceImage, int imgSize, int minFaceSize, int maxFaceSize);
```

Description

This function replaces existing face template at specified index in an user structure by a newly extracted face template from provided face image. Previously attached face template will be discarded.

Only faces with specific range of sizes are detected. The face size is defined as a maximum of inter pupil distance and distance between center of mouth and center point between eyes. The range is defined by minFaceSize and maxFaceSize parameters.

Each face contained in the user structure has its own index. Index numbers are given automatically when face templates are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int faceIndex`
[in] Index of face template that will be replaced

`const unsigned char * faceImage`
[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`int imgSize`
[in] size of the image in bytes.

`int minFaceSize`
[in] defines minimal expected face size in input image in pixels. If set to zero, default value is used (40). Minimum non-zero valid value is 12.

`int maxFaceSize`
[in] defines maximal expected face size in input image in pixels. If set to zero, default value is used (200).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter faceImage must not be NULL.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW (see page 24), IEngine_SetFaceRAW (see page 29), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.8 IEngine_SetFaceRelative Function

Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is relative to the image frame size.

C++

```
IDKIT_API int IEngine_SetFaceRelative(IENGINE_USER user, int faceIndex, const unsigned char
* faceImage, int imgSize, float minFaceSizeRatio, float maxFaceSizeRatio);
```

Description

This function replaces existing face template at specified index in an user structure by a newly extracted face template from

provided face image. Previously attached face template will be discarded.

Only faces with specific range of sizes are detected. The face size is defined as a maximum of inter pupil distance and distance between center of mouth and center point between eyes. The range is defined by minFaceSizeRatio and maxFaceSizeRatio parameters.

Each face contained in the user structure has its own index. Index numbers are given automatically when face templates are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int faceIndex`

[in] Index of face template that will be replaced

`const unsigned char * faceImage`

[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`int imgSize`

[in] size of the image in bytes.

`float minFaceSizeRatio`

[in] defines minimal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

`float maxFaceSizeRatio`

[in] defines maximal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter faceImage must not be NULL.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW (see page 24), IEngine_SetFaceRAW (see page 29), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.9 IEngine_SetFaceRAW Function

Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is absolute distance in pixels.

C++

```
IDKIT_API int IEngine_SetFaceRAW(IENGINE_USER user, int faceIndex, const unsigned char *
rawImage, int width, int height, int minFaceSize, int maxFaceSize);
```

Description

This function replaces existing face template at specified index in an user structure by a newly extracted face template from provided face image. Previously attached face template will be discarded.

Please see documentation of IEngine_SetFace (see page 27) and IEngine_AddFaceRAW (see page 24) for additional

details.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int faceIndex`

[in] Index of face template that will be replaced

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 24-bit (3 bytes per pixel) BGR format. The beginning is the upper left corner of the image. Value of black is (0x00,0x00,0x00), white is (0xFF,0xFF,0xFF).

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

`int minFaceSize`

[in] defines minimal expected face size in input image in pixels. If set to zero, default value is used (40). Minimum non-zero valid value is 12.

`int maxFaceSize`

[in] defines maximal expected face size in input image in pixels. If set to zero, default value is used (200).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW (see page 24), IEngine_SetFaceRAW, IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.10 IEngine_SetFaceRAWRelative Function

Performs face detection and extraction of face template from face image and replaces existing face template at specified index. Face image is read from memory buffer. Expected face size is relative to the image frame size.

C++

```
IDKIT_API int IEngine_SetFaceRAWRelative(IENGINE_USER user, int faceIndex, const unsigned
char * rawImage, int width, int height, float minFaceSizeRatio, float maxFaceSizeRatio);
```

Description

This function replaces existing face template at specified index in an user structure by a newly extracted face template from provided face image. Previously attached face template will be discarded.

Please see documentation of IEngine_SetFaceRelative (see page 28) and IEngine_AddFaceRAWRelative (see page 25) for additional details.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int faceIndex`

[in] Index of face template that will be replaced

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 24-bit (3 bytes per pixel) BGR format. The beginning is the upper left corner of the image. Value of black is (0x00,0x00,0x00), white is (0xFF,0xFF,0xFF).

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

`float minFaceSizeRatio`

[in] defines minimal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

`float maxFaceSizeRatio`

[in] defines maximal expected face size in input image. It is relative to the maximum of the input image dimensions. Minimum value is greater than 0, maximum is 1.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable face.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.

Related Topics

IEngine_AddFace (see page 22), IEngine_AddFaceRAW (see page 24), IEngine_SetFaceRAW (see page 29), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.11 IEngine_AddFaceTemplate Function

Adds existing face template to user structure. Face template is read from memory buffer.

C++

```
IDKIT_API int IEngine_AddFaceTemplate(IENGINE_USER user, const unsigned char *
templateData, int length);
```

Description

This function adds an additional face template to an initialized user structure.

Please see documentation of IEngine_AddFace (see page 22) for more details.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const unsigned char * templateData`

[in] Pointer to a face template stored in memory buffer. It has to be face template extracted using IDKit Multi SDK or using Innovatrics IFace SDK.

`int length`

[in] size of the face template in bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.

IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADTEMPLATE	Invalid template data.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter templateData must not be NULL.

Related Topics

IEngine_GetFaceTemplate (see page 92), IEngine_SetFaceTemplate (see page 32), IEngine_AddFace (see page 22), IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.12 IEngine_SetFaceTemplate Function

Replaces existing face template at specified index. Face template is read from memory buffer.

C++

```
IDKIT_API int IEngine_SetFaceTemplate(IENGINE_USER user, int faceIndex, const unsigned char * templateData, int length);
```

Description

This function replaces existing face template at specified index in an user structure by another face template. Previously attached face template will be discarded.

Please see documentation of IEngine_SetFace (see page 27) and IEngine_AddFaceTemplate (see page 31) for additional details.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int faceIndex`
[in] Index of face template that will be replaced

`const unsigned char * templateData`
[in] Pointer to a face template stored in memory buffer. It has to be face template extracted using IDKit Multi SDK or using Innovatrics IFace SDK.

`int length`
[in] size of the face template in bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_BADTEMPLATE	Invalid template data.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter templateData must not be NULL.

Related Topics

IEngine_GetFaceTemplate (see page 92), IEngine_SetFaceTemplate, IEngine_AddFace (see page 22), IEngine_SetFace (see page 27), IEngine_RemoveFace (see page 33), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.2.13 IEngine_RemoveFace Function

Removes face template at specified index

C++

```
IDKIT_API int IEngine_RemoveFace(IENGINE_USER user, int faceIndex);
```

Description

This function removes one of the face templates contained in input user structure.

Face template that will be removed is defined by index parameter. Each face template contained in the user structure has its own index. Removal of one face updates all remaining face template indices which are greater than the index of removed face template in such way, that indices are maintained in continuously ascending order. For instance, if a user structure has three faces with respective indices 0, 1, 2 and a face with index 1 is removed, the user structure will contain two faces with respective indices 0, 1. Index of a face with original value 2 was changed to have a value 1. Index numbers are given automatically when faces are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added face template is 0).

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int faceIndex`

[in] Index of the face template that will be removed

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFace (see page 22), IEngine_GetFaceCount (see page 46)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3 Add/Remove Iris

Functions

	Name	Description
	IEngine_AddIris (see page 34)	Performs extraction of iris template from iris image and adds it to user structure. Iris image is read from memory buffer.
	IEngine_AddIrisRAW (see page 35)	Performs extraction of iris template from iris RAW image and adds it to user structure. Iris RAW image is read from memory buffer.
	IEngine_AddIrisFromFile (see page 35)	Performs extraction of iris template from iris image and adds it to user structure. Iris image is read from file.
	IEngine_SetIris (see page 36)	Performs extraction of iris template from iris image and replaces existing iris template at specified index. Iris image is read from memory buffer.
	IEngine_SetIrisRAW (see page 37)	Performs extraction of iris template from iris image and replaces existing iris template at specified index. Iris image is read from memory buffer.
	IEngine_AddIrisTemplate (see page 38)	Adds existing iris template to user structure. Iris template is read from memory buffer.
	IEngine_SetIrisTemplate (see page 39)	Replaces existing iris template at specified index. Iris template is read from memory buffer.
	IEngine_RemoveIris (see page 40)	Removes iris template at specified index

2.1.3.2.3.1 IEngine_AddIris Function

Performs extraction of iris template from iris image and adds it to user structure. Iris image is read from memory buffer.

C++

```
IDKIT_API int IEngine_AddIris(IENGINE_USER user, IENGINE_IRIS_POSITION position,
IENGINE_IRIS_IMAGE_TYPE imageType, const unsigned char * irisImage, int imgSize, int
expectedIrisRadius);
```

Description

This function extracts and adds an additional iris template to an initialized user structure. You may add multiple irises to one user. You may even associate more iris images issued from the same iris to the same user. This helps to improve the recognition accuracy. Even if some of the iris templates are not recognized due to high noise or deformation, other samples of the same iris may still be found. This feature is also useful for multi-iris matching/identification. If you use two iris templates in one user for identification instead of one iris template, the error rate will be lower.

Input iris image is directly read from memory.

Each iris template contained in user structure has its own index. Index numbers are given automatically in ascending order, beginning at 0 (index of the first added iris is 0).

Note: If your input image is stored in RAW format (1 byte per pixel in grayscale), please use IEngine_AddIrisRAW (see page 35).

When you call this function, the input image is processed and the iris template is extracted from the original image.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`IENGINE_IRIS_POSITION position`
[in] Position of the current iris according to IENGINE_IRIS_POSITION (see page 111) enum

`IENGINE_IRIS_IMAGE_TYPE imageType`
[in] Type of input image according to IENGINE_IRIS_IMAGE_TYPE (see page 111) enum

`const unsigned char * irisImage`
[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`int imgSize`
[in] size of the image in bytes.

`int expectedIrisRadius`
[in] expected iris radius in pixels in input image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable iris. See CFG_IFACE_IRIS_DETECTION_THRESHOLD.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter irisImage must not be NULL.

Related Topics

IEngine_AddIrisRAW (see page 35), IEngine_AddIrisFromFile (see page 35), IEngine_SetIris (see page 36), IEngine_RemoveIris (see page 40), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3.2 IEngine_AddIrisRAW Function

Performs extraction of iris template from iris RAW image and adds it to user structure. Iris RAW image is read from memory buffer.

C++

```
IDKIT_API int IEngine_AddIrisRAW(IENGINE_USER user, IENGINE_IRIS_POSITION position,
IENGINE_IRIS_IMAGE_TYPE imageType, const unsigned char * rawImage, int width, int height,
int expectedIrisRadius);
```

Description

Please see documentation of IEngine_AddIris (see page 34) for more details.

Note: RAW image is RAW color 3 channels image. The rawImage is array of bytes where each pixels is formed by 3 bytes, which means 3 color channels ordered in BGR order. Origin of image is in upper-left corner. Size of the image data array can be calculated as image height *image width * 3 bytes.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`IENGINE_IRIS_POSITION position`

[in] Position of the current iris according to IENGINE_IRIS_POSITION (see page 111) enum

`IENGINE_IRIS_IMAGE_TYPE imageType`

[in] Type of input image according to IENGINE_IRIS_IMAGE_TYPE (see page 111) enum

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 24-bit (3 bytes per pixel) BGR format. The beginning is the upper left corner of the image. Value of black is (0x00,0x00,0x00), white is (0xFF,0xFF,0xFF).

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

`int expectedIrisRadius`

[in] expected iris radius in pixels in input image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable iris. See CFG_IFACE_IRIS_DETECTION_THRESHOLD.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.

Related Topics

IEngine_AddIris (see page 34), IEngine_AddIrisRAW, IEngine_SetIris (see page 36), IEngine_RemoveIris (see page 40), IEngine_InitUser (see page 12), IEngine_ConvertRawToImage (see page 89)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3.3 IEngine_AddIrisFromFile Function

Performs extraction of iris template from iris image and adds it to user structure. Iris image is read from file.

C++

```
IDKIT_API int IEngine_AddIrisFromFile(IENGINE_USER user, IENGINE_IRIS_POSITION position,
```

```
IENGINE_IRIS_IMAGE_TYPE imageType, const char * filename, int expectedIrisRadius);
```

Description

Please see documentation of IEngine_AddIris (see page 34) for more details.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`IENGINE_IRIS_POSITION position`
[in] Position of the current iris according to IENGINE_IRIS_POSITION (see page 111) enum

`IENGINE_IRIS_IMAGE_TYPE imageType`
[in] Type of input image according to IENGINE_IRIS_IMAGE_TYPE (see page 111) enum

`const char * filename`
[in] Name of the file containing iris image. Image format must be one of supported image formats (Image Processing (see page 6)).

`int expectedIrisRadius`
[in] expected iris radius in pixels in input image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable iris. See CFG_IFACE_IRIS_DETECTION_THRESHOLD.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter filename must not be NULL.

Related Topics

IEngine_AddIris (see page 34), IEngine_AddIrisRAW (see page 35), IEngine_SetIris (see page 36), IEngine_RemoveIris (see page 40), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3.4 IEngine_SetIris Function

Performs extraction of iris template from iris image and replaces existing iris template at specified index. Iris image is read from memory buffer.

C++

```
IDKIT_API int IEngine_SetIris(IENGINE_USER user, int irisIndex, IENGINE_IRIS_POSITION
position, IENGINE_IRIS_IMAGE_TYPE imageType, const unsigned char * irisImage, int imgSize,
int expectedIrisRadius);
```

Description

This function replaces existing iris template at specified index in an user structure by a newly extracted iris template from provided iris image. Previously attached iris template will be discarded.

Each iris contained in the user structure has its own index. Index numbers are given automatically when iris templates are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added fingerprint is 0).

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int irisIndex`
[in] Index of iris template that will be replaced

`IENGINE_IRIS_POSITION position`
[in] Position of the current iris according to IENGINE_IRIS_POSITION (see page 111) enum

`IENGINE_IRIS_IMAGE_TYPE imageType`
[in] Type of input image according to IENGINE_IRIS_IMAGE_TYPE (see page 111) enum

`const unsigned char * irisImage`
[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`int imgSize`
[in] size of the image in bytes.

`int expectedIrisRadius`
[in] expected iris radius in pixels in input image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable iris. See CFG_IFACE_IRIS_DETECTION_THRESHOLD.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter irisImage must not be NULL.

Related Topics

IEngine_AddIris (see page 34), IEngine_AddIrisRAW (see page 35), IEngine_SetIrisRAW (see page 37), IEngine_RemoveIris (see page 40), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3.5 IEngine_SetIrisRAW Function

Performs extraction of iris template from iris image and replaces existing iris template at specified index. Iris image is read from memory buffer.

C++

```
IDKIT_API int IEngine_SetIrisRAW(IENGINE_USER user, int irisIndex, IENGINE_IRIS_POSITION position, IENGINE_IRIS_IMAGE_TYPE imageType, const unsigned char * rawImage, int width, int height, int expectedIrisRadius);
```

Description

This function replaces existing iris template at specified index in an user structure by a newly extracted iris template from provided iris image. Previously attached iris template will be discarded.

Please see documentation of IEngine_SetIris (see page 36) and IEngine_AddIrisRAW (see page 35) for additional details.

Note: RAW image is RAW color 3 channels image. The rawImage is array of bytes where each pixels is formed by 3 bytes, which means 3 color channels ordered in BGR order. Origin of image is in upper-left corner. Size of the image data array can be calculated as image height *image width * 3 bytes.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int irisIndex`
[in] Index of iris template that will be replaced

`IENGINE_IRIS_POSITION position`
[in] Position of the current iris according to IENGINE_IRIS_POSITION (see page 111) enum

`IENGINE_IRIS_IMAGE_TYPE imageType`

[in] Type of input image according to IENGINE_IRIS_IMAGE_TYPE (see page 111) enum

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 24-bit (3 bytes per pixel) BGR format. The beginning is the upper left corner of the image. Value of black is (0x00,0x00,0x00), white is (0xFF,0xFF,0xFF).

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

`int expectedIrisRadius`

[in] expected iris radius in pixels in input image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable iris. See CFG_IFACE_IRIS_DETECTION_THRESHOLD.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.

Related Topics

IEngine_AddIris (see page 34), IEngine_AddIrisRAW (see page 35), IEngine_SetIrisRAW, IEngine_RemoveIris (see page 40), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3.6 IEngine_AddIrisTemplate Function

Adds existing iris template to user structure. Iris template is read from memory buffer.

C++

```
IDKIT_API int IEngine_AddIrisTemplate(IENGINE_USER user, IENGINE_IRIS_POSITION position,
const unsigned char * templateData, int length);
```

Description

This function adds an additional iris template to an initialized user structure.

Please see documentation of IEngine_AddIris (see page 34) for more details.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`IENGINE_IRIS_POSITION position`

[in] Position of the current iris according to IENGINE_IRIS_POSITION (see page 111) enum

`const unsigned char * templateData`

[in] Pointer to a iris template stored in memory buffer. It has to be iris template extracted using IDKit Multi SDK

`int length`

[in] size of the iris template in bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.

IENGINE_E_BADTEMPLATE	Invalid template data.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter templateData must not be NULL.

Related Topics

IEngine_GetIrisTemplate (see page 100), IEngine_SetIrisTemplate (see page 39), IEngine_AddIris (see page 34), IEngine_SetIris (see page 36), IEngine_RemoveIris (see page 40), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3.7 IEngine_SetIrisTemplate Function

Replaces existing iris template at specified index. Iris template is read from memory buffer.

C++

```
IDKIT_API int IEngine_SetIrisTemplate(IENGINE_USER user, int irisIndex,
IENGINE_IRIS_POSITION position, const unsigned char * templateData, int length);
```

Description

This function replaces existing iris template at specified index in an user structure by another iris template. Previously attached iris template will be discarded.

Please see documentation of IEngine_SetIris (see page 36) for additional details.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int irisIndex`
[in] Index of iris template that will be replaced

`IENGINE_IRIS_POSITION position`
[in] Position of the current iris according to IENGINE_IRIS_POSITION (see page 111) enum

`const unsigned char * templateData`
[in] Pointer to a iris template stored in memory buffer. It has to be iris template extracted using IDKit Multi SDK.

`int length`
[in] size of the iris template in bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_BADTEMPLATE	Invalid template data.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter templateData must not be NULL.

Related Topics

IEngine_GetIrisTemplate (see page 100), IEngine_SetIrisTemplate, IEngine_AddIris (see page 34), IEngine_SetIris (see page 36), IEngine_RemoveIris (see page 40), IEngine_InitUser (see page 12)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.2.3.8 IEngine_RemoveIris Function

Removes iris template at specified index

C++

```
IDKIT_API int IEngine_RemoveIris(IENGINE_USER user, int irisIndex);
```

Description

This function removes one of the iris templates contained in input user structure.

Iris template that will be removed is defined by index parameter. Each iris template contained in the user structure has its own index. Removal of one iris updates all remaining iris template indices which are greater than the index of removed iris template in such way, that indices are maintained in continuously ascending order. For instance, if a user structure has three irises with respective indices 0, 1, 2 and a iris with index 1 is removed, the user structure will contain two irises with respective indices 0, 1. Index of a iris with original value 2 was changed to have a value 1. Index numbers are given automatically when irises are added to the user structure. Index numbers are given in ascending order, beginning at 0 (the index of the first added iris template is 0).

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int irisIndex`

[in] Index of the iris template that will be removed

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddIris (see page 34), IEngine_GetIrisCount (see page 46)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.3 Properties

Functions

	Name	Description
	IEngine_SetCustomData (see page 48)	Attaches custom data string (see page 121) to the user.
	IEngine_GetCustomData (see page 49)	Retrieves custom data string (see page 121) attached to the user.

2.1.3.3.1 Fingerprint Properties

Functions

	Name	Description
	IEngine_GetFingerprintCount (see page 41)	Returns total number of fingerprints contained in user structure
	IEngine_GetFingerPosition (see page 41)	Returns finger position of fingerprint at specified index
	IEngine_SetFingerPosition (see page 42)	Sets finger position of fingerprint at specified index

	IEngine_FingerprintImageExists (see page 42)	Checks if fingerprint image with specified index is present in user record
	IEngine_GetFingerprintImage (see page 43)	Returns fingerprint image from user structure, image data is written to memory
	IEngine_AttachFingerprintImage (see page 44)	Links fingerprint image with a particular fingerprint
	IEngine_SaveFingerprintImage (see page 45)	Returns fingerprint image from user structure, saves the result to a file

2.1.3.3.1.1 IEngine_GetFingerprintCount Function

Returns total number of fingerprints contained in user structure

C++

```
IDKIT_API int IEngine_GetFingerprintCount(const IENGINE_USER user, int * fingerprintCount);
```

Description

This function returns total number of fingerprints contained in input user structure.

Each fingerprint contained in user structure has its own index. Index numbers are given automatically when fingerprints are added to user structure. Index numbers are given in ascending order, beginning at 0 (index of the first added fingerprint is 0).

Note: User structure parameter will not be modified by calling this function.

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int * fingerprintCount`
[out] On return, contains total number of fingerprints contained in user structure

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFingerprint (see page 14)

2.1.3.3.1.2 IEngine_GetFingerPosition Function

Returns finger position of fingerprint at specified index

C++

```
IDKIT_API int IEngine_GetFingerPosition(const IENGINE_USER user, int fingerprintIndex,
IENGINE_FINGER_POSITION * fingerPosition);
```

Description

This function returns finger position of fingerprint contained in input user structure. The index of the fingerprint which finger position will be returned is specified by fingerprintIndex parameter.

Note: User structure parameter will not be modified by calling this function.

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of the fingerprint which finger position will be returned

`IENGINE_FINGER_POSITION * fingerPosition`
[out] On return, contains finger position of specified fingerprint

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_GetFingerprintCount (see page 41), IEngine_SetFingerPosition (see page 42)

2.1.3.3.1.3 IEngine_SetFingerPosition Function

Sets finger position of fingerprint at specified index

C++

```
IDKIT_API int IEngine_SetFingerPosition(IENGINE_USER user, int fingerprintIndex,
IENGINE_FINGER_POSITION fingerPosition);
```

Description

This function sets finger position of the fingerprint contained in input user structure. The index of the fingerprint which finger position will be set is specified by the fingerprintIndex parameter.

Parameters

`IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of the fingerprint which finger position will be set

`IENGINE_FINGER_POSITION fingerPosition`
[in] New finger position

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADVALUE	Invalid fingerprintPosition value provided.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_GetFingerprintCount (see page 41), IEngine_GetFingerPosition (see page 41)

2.1.3.3.1.4 IEngine_FingerprintImageExists Function

Checks if fingerprint image with specified index is present in user record

C++

```
IDKIT_API int IEngine_FingerprintImageExists(const IENGINE_USER user, int index, int *
exists);
```

Description

This function checks fingerprint user structer cotains fingeprint image on specified index

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int * exists`
[out] On return, contains 1 if user has fingerprint with specified index, otherwise 0

`fingerprintIndex`
[in] Index of the fingerprint which finger position will be checked

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_GetFingerprintCount (see page 41), IEngine_GetFingerprintImage (see page 43)

2.1.3.3.1.5 IEngine_GetFingerprintImage Function

Returns fingerprint image from user structure, image data is written to memory

C++

```
IDKIT_API int IEngine_GetFingerprintImage(const IENGINE_USER user, int fingerprintIndex,
IENGINE_IMAGE_FORMAT format, unsigned char * fingerprintImage, int * length);
```

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of the fingerprint which image will be returned

`IENGINE_IMAGE_FORMAT format`
[in] Format of the output image

`unsigned char * fingerprintImage`
[out] Pointer to the memory space where image will be saved

`int * length`
[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the fingerprintImage parameter. On return, this parameter will be equal to the total image length.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_NOIMAGE	Image not available. (This may be returned if user record was loaded from database and if images are not automatically saved in the database, see CFG_STORE_IMAGES (see page 103) for more details.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function retrieves fingerprint image from input user structure. The index of the fingerprint image that will be returned is specified by the fingerprintIndex parameter. Resulting image data is written to memory. This function uses two-phase buffer retrieval (see page 118) algorithm.

Example

```
...
IENGINE_USER user=IEngine_InitUser();
IEngine_AddFingerprintFromFile(user,0,"image.bmp");
int length=0;
IEngine_GetFingerprintImage(user, 0, IENGINE_FORMAT_WSQ, NULL, &length);
printf("Total image length is %d bytes.\n",length);

unsigned char *imageData=new unsigned char[length];
int err=IEngine_GetFingerprintImage(user, 0, IENGINE_FORMAT_WSQ, imageData, &length);
if(err!=0)
    printf("Image.bmp was just written in WSQ format to imageData\n");
else
    printf("Error occurred\n");
...
delete[] imageData;
IEngine_FreeUser(user);
```

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_SaveFingerprintImage (see page 45), IEngine_GetFingerprintCount (see page 41)

2.1.3.3.1.6 IEngine_AttachFingerprintImage Function

Links fingerprint image with a particular fingerprint

C++

```
IDKIT_API int IEngine_AttachFingerprintImage(IENGINE_USER user, int fingerprintIndex, const unsigned char * fingerprintImage);
```

Description

This function links the input fingerprint image with a fingerprint contained within the user structure. The fingerprint concerned by this function is defined by the fingerprintIndex parameter. As opposed to IEngine_SetFingerprint (see page 18) function, this function does not alter a previously generated fingerprint template or the finger position value associated with the concerned fingerprint.

This function is supposed to be used in combination with IEngine_ImportUserTemplate (see page 57) in order to attach images to imported fingerprint templates.

The potential risk, when using this function improperly, is that fingerprint images will be linked to non-corresponding fingerprint templates.

Note: If your fingerprint image is stored in raw format, please use IEngine_ConvertRawToImage (see page 89) before calling this function

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int fingerprintIndex`

[in] Index of fingerprint which image will be modified

`const unsigned char * fingerprintImage`

[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_NULLPARAM	Parameter fingerprintImage must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.

IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_MEMORY	Memory allocation failed.

Related Topics

IEngine_ImportUserTemplate (see page 57), IEngine_SetFingerprint (see page 18), IEngine_ConvertRawToImage (see page 89)

2.1.3.3.1.7 IEngine_SaveFingerprintImage Function

Returns fingerprint image from user structure, saves the result to a file

C++

```
IDKIT_API int IEngine_SaveFingerprintImage(const IENGINE_USER user, int fingerprintIndex,
IENGINE_IMAGE_FORMAT format, const char * filename);
```

Description

This function retrieves fingerprint image from input user structure. The index of the fingerprint image that will be returned is specified by the fingerprintIndex parameter. Resulting image is saved to a file.

Note: User structure parameter will not be modified by calling this function.

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of the fingerprint which image will be returned

`IENGINE_IMAGE_FORMAT format`
[in] Format of the output image

`const char * filename`
[in] Name of the file where the returned image will be stored.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_NOIMAGE	Image not available. (This may be returned if user record was loaded from database and if images are not automatically saved in the database, see CFG_STORE_IMAGES (see page 103) for more details.
IENGINE_E_FILE	Error occurred while opening/reading file.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter filename must not be NULL.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_GetFingerprintImage (see page 43), IEngine_GetFingerprintCount (see page 41)

2.1.3.3.2 Face Properties

Functions

	Name	Description
	IEngine_GetFaceCount (see page 46)	Returns total number of face templates contained in user structure.

2.1.3.3.2.1 IEngine_GetFaceCount Function

Returns total number of face templates contained in user structure.

C++

```
IDKIT_API int IEngine_GetFaceCount(const IENGINE_USER user, int * count);
```

Description

Each face template contained in user structure has its own index. Index numbers are given automatically when faces are added to user structure. Index numbers are given in ascending order, beginning at 0 (index of the first added face is 0).

Note: User structure parameter will not be modified by calling this function.

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int * count`
[out] On return, contains total number of faces contained in user structure

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddFace (see page 22), IEngine_RemoveFace (see page 33)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.3.3 Iris Properties

Functions

	Name	Description
	IEngine_GetIrisCount (see page 46)	Returns total number of iris templates contained in user structure.
	IEngine_GetIrisPosition (see page 47)	Returns iris position of iris at specified index
	IEngine_SetIrisPosition (see page 47)	Sets iris position of iris at specified index

2.1.3.3.3.1 IEngine_GetIrisCount Function

Returns total number of iris templates contained in user structure.

C++

```
IDKIT_API int IEngine_GetIrisCount(const IENGINE_USER user, int * count);
```

Description

Each iris template contained in user structure has its own index. Index numbers are given automatically when irises are added to user structure. Index numbers are given in ascending order, beginning at 0 (index of the first added iris is 0).

Note: User structure parameter will not be modified by calling this function.

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int * count`

[out] On return, contains total number of irises contained in user structure

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddIris (see page 34), IEngine_RemoveIris (see page 40)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.3.3.2 IEngine_GetIrisPosition Function

Returns iris position of iris at specified index

C++

```
IDKIT_API int IEngine_GetIrisPosition(const IENGINE_USER user, int irisIndex,
IENGINE_IRIS_POSITION * irisPosition);
```

Description

This function returns iris position of iris contained in input user structure. The index of the iris which iris position will be returned is specified by irisIndex parameter.

Note: User structure parameter will not be modified by calling this function.

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int irisIndex`
[in] Index of the iris which iris position will be returned

`IENGINE_IRIS_POSITION * irisPosition`
[out] On return, contains iris position, see IENGINE_IRIS_POSITION (see page 111) for possible values

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_NOIRIS	User structure contains no iris templates.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddIris (see page 34), IEngine_GetIrisCount (see page 46), IEngine_SetIrisPosition (see page 47)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK..

2.1.3.3.3.3 IEngine_SetIrisPosition Function

Sets iris position of iris at specified index

C++

```
IDKIT_API int IEngine_SetIrisPosition(IENGINE_USER user, int irisIndex,
IENGINE_IRIS_POSITION irisPosition);
```

Description

This function sets iris position of the iris contained in input user structure. The index of the iris which iris position will be set is specified by the irisIndex parameter.

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int irisIndex`

[in] Index of the iris which iris position will be set

`IENGINE_IRIS_POSITION irisPosition`

[in] New iris position, see IENGINE_IRIS_POSITION (see page 111) for possible values

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_NOIRIS	User structure contains no iris templates.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_AddIris (see page 34), IEngine_GetIrisCount (see page 46), IEngine_GetIrisPosition (see page 47)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK..

2.1.3.3.4 IEngine_SetCustomData Function

Attaches custom data string (see page 121) to the user.

C++

```
IDKIT_API int IEngine_SetCustomData(IENGINE_USER user, const unsigned char * data, int length);
```

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`const unsigned char * data`

[in] Pointer to application specific data that will be attached to the specified user. This parameter may be NULL.

`int length`

[in] Total length of the memory space pointed by data parameter.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_GetCustomData (see page 49)

2.1.3.3.5 IEngine_GetCustomData Function

Retrieves custom data string (see page 121) attached to the user.

C++

```
IDKIT_API int IEngine_GetCustomData(const IENGINE_USER user, unsigned char * data, int * length);
```

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`unsigned char * data`

[out] Pointer to memory space where application specific data will be written.

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the data parameter. On return, this parameter will be equal to the total length of application specific data attached to the user.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

Custom data string is retrieved together with user record when IEngine_GetUser (see page 69) is called. This function copies the custom data string from user record to the provided buffer. This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_SetCustomData (see page 48)

2.1.3.4 Biometry

Functions

	Name	Description
	IEngine_GetFingerprintPresence (see page 50)	This function returns presence quality of the input fingerprint image. Image quality number is calculated as an area where presence of fingerprint is detected. This function does not measure quality of the presented fingerprint pattern. Use IEngine_GetFingerprintQuality (see page 51) function to evaluate fingerprint pattern quality.
	IEngine_GetFingerprintPresenceRAW (see page 50)	This function returns presence quality of input raw fingerprint image. Image quality number is calculated as an area where presence of fingerprint is detected. This function does not measure quality of the presented fingerprint pattern. Use IEngine_GetFingerprintQuality (see page 51) function to evaluate fingerprint pattern quality.
	IEngine_GetFingerprintQuality (see page 51)	Returns quality of fingerprint at specified index.
	IEngine_GetDeltasAndCores (see page 51)	Returns list of fingerprint critical points (core and deltas)
	IEngine_GetMinutiaeImage (see page 52)	Returns fingerprint minutiae points overlapped on the original fingerprint image, image is returned in memory
	IEngine_SaveMinutiaeImage (see page 53)	Saves fingerprint minutiae points overlapped on the original fingerprint image
	IEngine_GetFaceQuality (see page 54)	Returns quality of face at specified index. Face quality is a value ranging between 0 and 255 (0 - undefined, 1 - min quality, 255 - max quality).
	IEngine_GetIrisQuality (see page 54)	Returns quality of iris at specified index. Iris quality is a value ranging between 0 and 255 (0 - undefined, 1 - min quality, 255 - max quality).

2.1.3.4.1 IEngine_GetFingerprintPresence Function

This function returns presence quality of the input fingerprint image. Image quality number is calculated as an area where presence of fingerprint is detected. This function does not measure quality of the presented fingerprint pattern. Use IEngine_GetFingerprintQuality (see page 51) function to evaluate fingerprint pattern quality.

C++

```
IDKIT_API int IEngine_GetFingerprintPresence(const unsigned char * fingerprintImage, int imgSize, int * presence);
```

Parameters

`const unsigned char * fingerprintImage`

[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)). Minimal allowed size of image is 90x90 pixels and maximal size of image is 4000x4000 pixels.

`int imgSize`

[in] length of fingerprintImage buffer

`int * presence`

[out] On return, contains measure of fingerprint presence on the fingerprint scanner surface calculated from the input fingerprint image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable fingerprint.
IENGINE_E_SMALLIMAGE	Image is too small. Minimal size of image must be 90x90.
IENGINE_E_BIGIMAGE	Image is too big. Maximal size of image must be 4000x4000.
IENGINE_E_NULLPARAM	Parameter fingerprintImage must not be NULL.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This method is similar to method IEngine_GetImageQuality in Innovatrics ANSI&ISO SDK and ISegLib_GetImageQuality in Segmentation SDK.

Related Topics

IEngine_GetFingerprintPresenceRAW (see page 50), IEngine_GetFingerprintQuality (see page 51), IEngine_GetFingerprintClass (see page 99)

2.1.3.4.2 IEngine_GetFingerprintPresenceRAW Function

This function returns presence quality of input raw fingerprint image. Image quality number is calculated as an area where presence of fingerprint is detected. This function does not measure quality of the presented fingerprint pattern. Use IEngine_GetFingerprintQuality (see page 51) function to evaluate fingerprint pattern quality.

C++

```
IDKIT_API int IEngine_GetFingerprintPresenceRAW(const unsigned char * rawImage, int width, int height, int * presence);
```

Parameters

`const unsigned char * rawImage`

[in] Pointer to a raw image stored in memory. The image must be of no-header uncompressed raw 8-bit (one byte per pixel) grayscale format. The beginning is the upper left corner of the image. Value of black is 0x0, white is 0xFF. Minimal allowed size of image is 90x90 pixels and maximal size of image is 4000x4000 pixels.

`int width`

[in] The number of pixels indicating the width of the raw image

`int height`

[in] The number of pixels indicating the height of the raw image

`int * presence`
[out] On return, contains measure of fingerprint presence on the fingerprint scanner surface calculated from the input fingerprint image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable fingerprint.
IENGINE_E_SMALLIMAGE	Image is too small. Minimal size of image must be 90x90.
IENGINE_E_BIGIMAGE	Image is too big. Maximal size of image must be 4000x4000.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	Parameter rawImage must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This method is similar to method IEngine_GetImageQuality in Innovatrics ANSI&ISO SDK and ISegLib_GetImageQuality in Segmentation SDK.

Related Topics

IEngine_GetFingerprintPresence (see page 50), IEngine_GetFingerprintQuality (see page 51), IEngine_GetFingerprintClass (see page 99)

2.1.3.4.3 IEngine_GetFingerprintQuality Function

Returns quality of fingerprint at specified index.

C++

```
IDKIT_API int IEngine_GetFingerprintQuality(const IENGINE_USER user, int fingerprintIndex,
int * quality);
```

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of the fingerprint which quality level will be returned

`int * quality`
[out] On return, contains quality of the specified fingerprint

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This is an Innovatrics-defined fingerprint quality (see page 131) of the fingerprint specified by fingerprintIndex parameter.

Related Topics

IEngine_AddFingerprint (see page 14), IEngine_GetFingerprintCount (see page 41)

2.1.3.4.4 IEngine_GetDeltasAndCores Function

Returns list of fingerprint critical points (core and deltas)

C++

```
IDKIT_API int IEngine_GetDeltasAndCores(const IENGINE_USER user, int fingerprintIndex, int
* criticalPointsCount, IENGINE_CRITICAL_POINT criticalPoints[MAX_CRITICAL_POINTS_COUNT]);
```

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`int fingerprintIndex`

[in] Index of the fingerprint which critical points list will be returned

`int * criticalPointsCount`

[out] On return, this parameter will contain the total number of extracted critical points.

`IENGINE_CRITICAL_POINT criticalPoints[MAX_CRITICAL_POINTS_COUNT]`

[out] On return, this array will contain complete list of extracted critical points. This parameter must be initialized on input, it should contain at least MAX_CRITICAL_POINTS_COUNT (see page 110) elements.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function returns list of critical points (core and deltas) for a specified fingerprint. Each critical point is represented by its location, angle and type. The index of the fingerprint image which minutiae are returned is specified by the fingerprintIndex parameter. Please notice that critical points are identified during fingerprint template extraction from fingerprint image.

Compatibility

Core and deltas extraction can be enabled/disabled through CFG_EXTRACT_CRITICAL_POINTS (see page 103) parameter. Core and deltas extraction is enabled by default except in IDKit Android SDK and IDKit Embedded SDK where it's disabled by default.

Related Topics

IEngine_GetMinutiaePoints (see page 101)

2.1.3.4.5 IEngine_GetMinutiaeImage Function

Returns fingerprint minutiae points overlapped on the original fingerprint image, image is returned in memory

C++

```
IDKIT_API int IEngine_GetMinutiaeImage(const IENGINE_USER user, int fingerprintIndex,
IENGINE_IMAGE_FORMAT format, unsigned char * minutiaeImage, int * length);
```

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`int fingerprintIndex`

[in] Index of the fingerprint which minutiae will be drawn

`IENGINE_IMAGE_FORMAT format`

[in] Format of the output image (use BMP for colorized image)

`unsigned char * minutiaeImage`

[out] Pointer to the memory space where image will be saved

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the minutiaeImage parameter. On return, this parameter will be equal to the total image length.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NOTCONNECTED	IDKit not connected.

Remarks

This function draws minutiae points on the original fingerprint image. The index of the fingerprint image is specified by the fingerprintIndex parameter. Resulting image data is written to memory. Minutia points are depicted as blue and red vectors. Ridge endings are blue and bifurcations are red.

Note: Use BMP format of output image in order to get colorized image. All the other output formats will be black and white.

Related Topics

IEngine_SaveMinutiaeImage (see page 53), IEngine_GetMinutiaePoints (see page 101)

2.1.3.4.6 IEngine_SaveMinutiaeImage Function

Saves fingerprint minutiae points overlapped on the original fingerprint image

C++

```
IDKIT_API int IEngine_SaveMinutiaeImage(const IENGINE_USER user, int fingerprintIndex,
IENGINE_IMAGE_FORMAT format, const char * filename);
```

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int fingerprintIndex`
[in] Index of the fingerprint which minutiae will be drawn

`IENGINE_IMAGE_FORMAT format`
[in] Format of the output image (use BMP for colorized image)

`const char * filename`
[in] Name of the input image

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_FILE	Erroroccurred while opening/writing file.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameter filename must not be NULL.

Remarks

This function draws minutiae points on the original fingerprint image. The index of the fingerprint image is specified by the fingerprintIndex parameter. Resulting image data is written to a file. Minutia points are depicted as blue and red vectors. Ridge endings are blue and bifurcations are red.

Note: Use BMP format of output image in order to get colorized image. All the other output formats will be black and white.

Related Topics

IEngine_SaveMinutiaeImage, IEngine_GetMinutiaePoints (see page 101)

2.1.3.4.7 IEngine_GetFaceQuality Function

Returns quality of face at specified index. Face quality is a value ranging between 0 and 255 (0 - undefined, 1 - min quality, 255 - max quality).

C++

```
IDKIT_API int IEngine_GetFaceQuality(const IENGINE_USER user, int faceIndex, int * quality);
```

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int faceIndex`
[in] Index of the face which quality level will be returned

`int * quality`
[out] On return, contains quality of the specified face

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_NOFACES	User structure contains no face templates.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This is an Innovatrics-defined face quality (see page 132) of the face specified by faceIndex parameter.

Related Topics

IEngine_AddFace (see page 22), IEngine_GetFaceCount (see page 46)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.4.8 IEngine_GetIrisQuality Function

Returns quality of iris at specified index. Iris quality is a value ranging between 0 and 255 (0 - undefined, 1 - min quality, 255 - max quality).

C++

```
IDKIT_API int IEngine_GetIrisQuality(const IENGINE_USER user, int irisIndex, int * quality);
```

Parameters

`const IENGINE_USER user`
[in] An initialized user structure

`int irisIndex`
[in] Index of the iris which quality level will be returned

`int * quality`
[out] On return, contains quality of the specified iris

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.

IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_NOIRIS	User structure contains no iris templates.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This is an Innovatrics-defined iris quality (see page 132) of the iris specified by irisIndex parameter.

Related Topics

IEngine_AddIris (see page 34), IEngine_GetIrisCount (see page 46)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.3.5 Import/Export

Functions

	Name	Description
	IEngine_SerializeUser (see page 55)	Converts user structure into a sequence of bytes
	IEngine_DeserializeUser (see page 56)	Converts a sequence of bytes into the corresponding user structure
	IEngine_ExportUserTemplate (see page 56)	Exports user template
	IEngine_ImportUserTemplate (see page 57)	Imports fingerprint, face and iris templates from user template

2.1.3.5.1 IEngine_SerializeUser Function

Converts user structure into a sequence of bytes

C++

```
IDKIT_API int IEngine_SerializeUser(const IENGINE_USER user, bool serializeImages, unsigned char * buffer, int * length);
```

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`bool serializeImages`

[in] Boolean indicating whether the resulting sequence of bytes should contain images that are (possibly) stored in the input user structure

`unsigned char * buffer`

[out] Pointer to the memory space where the output sequence of bytes will be written

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the buffer parameter. On return, this parameter will be equal to the total number of bytes in the resulting sequence.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function converts user structure into a corresponding sequence of bytes. This function is useful if you for example need to send user's information (biometric templates, images, custom data and tags) to a distant computer. This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_DeserializeUser (see page 56)

2.1.3.5.2 IEngine_DeserializeUser Function

Converts a sequence of bytes into the corresponding user structure

C++

```
IDKIT_API int IEngine_DeserializeUser(IENGINE_USER user, const unsigned char * buffer);
```

Description

This function converts a sequence of bytes into the corresponding user structure. The input sequence of bytes, in order to be valid, has to be created by a prior call of IEngine_SerializeUser (see page 55) function. All data (biometric templates and images) contained within the provided user structure will be lost. On return, the provided user structure will only contain data stored in the input sequence of bytes.

Parameters

`IENGINE_USER user`

[out] An initialized user structure

`const unsigned char * buffer`

[in] Pointer to memory space containing the input sequence of bytes

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADTEMPLATE	Input sequence of bytes doesn't represent a valid user stucture.
IENGINE_E_NULLPARAM	Parameter buffer must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_SerializeUser (see page 55)

2.1.3.5.3 IEngine_ExportUserTemplate Function

Exports user template

C++

```
IDKIT_API int IEngine_ExportUserTemplate(const IENGINE_USER user, IENGINE_TEMPLATE_FORMAT format, unsigned char * templateData, int * length);
```

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`IENGINE_TEMPLATE_FORMAT format`

[in] Format of the output user template

`unsigned char * templateData`

[out] Pointer to the memory space where template raw data will be saved

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by templateData. On return, this parameter will be equal to the total length of user template.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.

IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_NOTEMPLATES	User structure contains no templates (void user).
IENGINE_E_BADFORMAT	Unsupported user template format.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function exports user template as raw data. User template consists of all fingerprint, face and iris templates contained within the user structure. User template does not contain images, custom data and tags. It contains only biometric templates with its positions (fingerprint and iris position). It can be exported in one of the formats enumerated in IENGINE_TEMPLATE_FORMAT (see page 109). This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_ImportUserTemplate (see page 57)

2.1.3.5.4 IEngine_ImportUserTemplate Function

Imports fingerprint, face and iris templates from user template

C++

```
IDKIT_API int IEngine_ImportUserTemplate(IENGINE_USER user, IENGINE_TEMPLATE_FORMAT format,
const unsigned char * templateData);
```

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`IENGINE_TEMPLATE_FORMAT format`

[in] Format of the input user template.

`const unsigned char * templateData`

[in] Pointer to user template data to be imported into user structure

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NULLPARAM	Parameter templateData must not be NULL.
IENGINE_E_NOTEMPLATES	User structure contains no templates (void user).
IENGINE_E_BADFORMAT	User template format is not supported for import.
IENGINE_E_BADTEMPLATE	Invalid user template.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function imports all fingerprint, face and iris templates from input user template. Imported biometrics templates are added to the user structure. If the input user structure already contains any templates, imported templates are added at the end (at last position).

As the user template does not contain images, fingerprints, faces and irises imported to the user structure will not have corresponding images. In other words the available data about imported fingerprints, faces and irises will only consist of the biometric template and its position (fingerprint and iris position). However, if you need to link the fingerprint image to imported fingerprints, you may use function IEngine_AttachFingerprintImage (see page 44).

Note that not all template types can be imported. Check descriptions of template formats in IENGINE_TEMPLATE_FORMAT (see page 109) to find out which formats can be imported.

Related Topics

IEngine_ExportUserTemplate (see page 56), IEngine_AttachFingerprintImage (see page 44)

2.1.3.6 Tags

Functions

	Name	Description
	IEngine_SetIntTag (see page 58)	Sets tag (see page 135) value to the specified integer value.
	IEngine_GetIntTag (see page 59)	Gets tag (see page 135) value as an integer.
	IEngine_SetStringTag (see page 59)	Sets tag (see page 135) value to the specified string.
	IEngine_GetStringTag (see page 60)	Gets tag (see page 135) value as a string.
	IEngine_HasTag (see page 61)	Checks whether user has a tag (see page 135) set.
	IEngine_ClearTag (see page 61)	Removes tag (see page 135) from the user.
	IEngine_GetTagCount (see page 62)	Gets the total number of tags (see page 135) set on a user.
	IEngine_GetTagName (see page 62)	Gets name of a tag (see page 135) at the specified index.

2.1.3.6.1 IEngine_SetIntTag Function

Sets tag (see page 135) value to the specified integer value.

C++

```
IDKIT_API int IEngine_SetIntTag(IENGINE_USER user, const char * name, int value);
```

Parameters

`IENGINE_USER user`
[in] User whose tag is to be set.

`const char * name`
[in] Name of the tag to set.

`int value`
[in] New tag value.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameter name must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

If the tag doesn't exist, it is created. If the tag exists, it is overwritten. All tag values are internally strings. The integer value is automatically converted to its string representation.

Related Topics

IEngine_GetIntTag (see page 59), IEngine_SetStringTag (see page 59), IEngine_ClearTag (see page 61), IEngine_HasTag (see page 61)

2.1.3.6.2 IEngine_GetIntTag Function

Gets tag (see page 135) value as an integer.

C++

```
IDKIT_API int IEngine_GetIntTag(const IENGINE_USER user, const char * name, int * value);
```

Parameters

`const IENGINE_USER user`
[in] User that contains the tag.

`const char * name`
[in] Name of the tag to read.

`int * value`
[out] Tag value.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameters name and value must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

If the tag doesn't exist, this function returns zero. All tag values are internally strings. The tag value is automatically converted to integer. If the conversion fails, this function returns zero.

Related Topics

IEngine_SetIntTag (see page 58), IEngine_GetStringTag (see page 60). IEngine_HasTag (see page 61)

2.1.3.6.3 IEngine_SetStringTag Function

Sets tag (see page 135) value to the specified string.

C++

```
IDKIT_API int IEngine_SetStringTag(IENGINE_USER user, const char * name, const char *
value);
```

Parameters

`IENGINE_USER user`
[in] User whose tag is to be set.

`const char * name`
[in] Name of the tag to set.

`const char * value`
[in] String containing new tag value.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameters name and value must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

If the tag doesn't exist, it is created. If the tag exists, it is overwritten.

Related Topics

IEngine_GetStringTag (see page 60), IEngine_SetIntTag (see page 58), IEngine_ClearTag (see page 61), IEngine_HasTag (see page 61)

2.1.3.6.4 IEngine_GetStringTag Function

Gets tag (see page 135) value as a string.

C++

```
IDKIT_API int IEngine_GetStringTag(const IENGINE_USER user, const char * name, char *
value, int * length);
```

Parameters

`const IENGINE_USER user`
[in] User that contains the tag.

`const char * name`
[in] Name of the tag to read.

`char * value`
[out] Tag value.

`int * length`
[in/out] On input, this is the size of allocated buffer for tag value. On output, it contains length of the tag value plus 1 byte for null character.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameters name and length must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

If the tag doesn't exist, this function returns empty string. This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_SetStringTag (see page 59), IEngine_GetIntTag (see page 59), IEngine_HasTag (see page 61)

2.1.3.6.5 IEngine_HasTag Function

Checks whether user has a tag (see page 135) set.

C++

```
IDKIT_API int IEngine_HasTag(const IENGINE_USER user, const char * name, int * present);
```

Parameters

`const IENGINE_USER user`
[in] User to check.

`const char * name`
[in] Tag name to check.

`int * present`
[out] Value 1 if the tag is found. Value 0 if there is no such tag.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameter name must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_ClearTag (see page 61)

2.1.3.6.6 IEngine_ClearTag Function

Removes tag (see page 135) from the user.

C++

```
IDKIT_API int IEngine_ClearTag(IENGINE_USER user, const char * name);
```

Parameters

`IENGINE_USER user`
[in] User that contains the tag.

`const char * name`
[in] Name of the tag to remove.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameter name must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

If the tag doesn't exist, this function has no effect. If the tag exists, it is removed from the user record.

Related Topics

IEngine_HasTag (see page 61)

2.1.3.6.7 IEngine_GetTagCount Function

Gets the total number of tags (see page 135) set on a user.

C++

```
IDKIT_API int IEngine_GetTagCount(const IENGINE_USER user, int * count);
```

Parameters

`const IENGINE_USER user`
[in] User that contains the tags.

`int * count`
[out] Number of tags set on the user.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_GetTagName (see page 62), IEngine_HasTag (see page 61)

2.1.3.6.8 IEngine_GetTagName Function

Gets name of a tag (see page 135) at the specified index.

C++

```
IDKIT_API int IEngine_GetTagName(const IENGINE_USER user, int offset, char * name, int *
length);
```

Parameters

`const IENGINE_USER user`
[in] User containing the tag.

`int offset`
[in] Offset into an array of tags in range of 0 to N-1 where N is the number of tags in the user record. Tags have arbitrary order.

`char * name`
[out] Name of the tag.

`int * length`
[in/out] On input, this is the size of the buffer receiving the tag. On output, it contains actual length of the tag name plus 1 byte for null character.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function can be used to retrieve names of all tags set on a user. Application loops through offsets from 0 to number of tags minus one. Number of tags can be obtained from function IEngine_GetTagCount (see page 62). Tag names are returned in an arbitrary order. Adding, removing, or setting of tags changes the order. This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_GetTagCount (see page 62), IEngine_HasTag (see page 61), IEngine_GetStringTag (see page 60), IEngine_GetIntTag (see page 59)

2.1.4 Collections of Users

Functions

	Name	Description
	IEngine_InitCollection (see page 63)	Initializes collection of user IDs.
	IEngine_FreeCollection (see page 64)	Releases memory held by IENGINE_COLLECTION (see page 103).
	IEngine_GetCollectionSize (see page 64)	Gets the number of IDs in the collection.
	IEngine_GetCollectionIDs (see page 64)	Retrieves IDs from a collection.

2.1.4.1 IEngine_InitCollection Function

Initializes collection of user IDs.

C++

```
IDKIT_API IENGINE_COLLECTION IEngine_InitCollection();
```

Returns

If this function succeeds, it returns valid IENGINE_COLLECTION (see page 103). If it fails, NULL is returned.

Return Values

Return Values	Description
NULL	Memory error - could not allocate memory.
IENGINE_COLLECTION (see page 103)	An initialized empty IENGINE_COLLECTION (see page 103) structure.

Remarks

This function is used to construct new IENGINE_COLLECTION (see page 103) before it can be used by other functions. After use, memory can be released by calling IEngine_FreeCollection (see page 64).

Related Topics

IEngine_FreeCollection (see page 64), IENGINE_COLLECTION (see page 103)

2.1.4.2 IEngine_FreeCollection Function

Releases memory held by IENGINE_COLLECTION (see page 103).

C++

```
IDKIT_API int IEngine_FreeCollection(IENGINE_COLLECTION collection);
```

Parameters

`IENGINE_COLLECTION collection`

[in] Collection to be released.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NULLPARAM	Parameter collection must not be NULL.

Related Topics

IEngine_InitCollection (see page 63), IENGINE_COLLECTION (see page 103)

2.1.4.3 IEngine_GetCollectionSize Function

Gets the number of IDs in the collection.

C++

```
IDKIT_API int IEngine_GetCollectionSize(const IENGINE_COLLECTION collection, int * count);
```

Parameters

`const IENGINE_COLLECTION collection`

[in] Size of this collection is returned.

`int * count`

[out] Number of user IDs in the collection.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameters count and collection must not be NULL.
IENGINE_E_BADVALUE	Invalid value provided.

2.1.4.4 IEngine_GetCollectionIDs Function

Retrieves IDs from a collection.

C++

```
IDKIT_API int IEngine_GetCollectionIDs(const IENGINE_COLLECTION collection, int * ids, int
count);
```

Parameters

`const IENGINE_COLLECTION collection`
[in] Collection. IDs are copied from this collection.

`int * ids`
[out] Array that receives the list of IDs.

`int count`
[in] Number of allocated items in ids array.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NULLPARAM	Parameters count and collection must not be NULL.
IENGINE_E_BADVALUE	Invalid value provided.

Remarks

This function copies all IDs (or up to the capacity of ids array) from IENGINE_COLLECTION (see page 103) into application-provided array.

Related Topics

IEngine_GetCollectionSize (see page 64)

2.1.5 Database

Functions

	Name	Description
	IEngine_RegisterUser (see page 65)	Adds user to database
	IEngine_RegisterUserAs (see page 66)	Adds user to database and sets its id number
	IEngine_UpdateUser (see page 67)	Updates user data contained in database
	IEngine_UserExists (see page 68)	Checks if user already exists in the database
	IEngine_GetUserIfExists (see page 68)	Checks if user already exists in the database and retrieves user data from database
	IEngine_GetUser (see page 69)	Retrieves user data from database
	IEngine_RemoveUser (see page 69)	Removes user from database
	IEngine_ClearDatabase (see page 70)	Deletes all entries from database
	IEngine_GetUserCount (see page 70)	Returns total number of users registered in database
	IEngine_GetAllUserIDs (see page 71)	Retrieves list of all user IDs in the database.
	IEngine_GetUserIDsByQuery (see page 71)	Retrieves list of user IDs in the database that match tag query (see page 133).

2.1.5.1 IEngine_RegisterUser Function

Adds user to database

C++

```
IDKIT_API int IEngine_RegisterUser(const IENGINE_USER user, int * userID);
```

Description

This function registers a previously initialized user in the currently open database. The user id is automatically allocated by the library, guaranteeing the uniqueness of user ids. Valid user id is a positive integer (greater or equal to 1).

Parameters

`const IENGINE_USER user`

[in] Contains user data to be added to database

`int * userID`

[out] On return, contains userID attributed to newly registered user

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_DBFULL	Exceeded database user limit.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Example

```
...
int err;
int userID;
IENGINE_USER user=IEngine_InitUser();
IEngine_AddFingerprintFromFile(user,0,"image.bmp");
err=IEngine_RegisterUser(user, &userID);
if(err!=0)
        printf("Error occurred during registration\n");
else
        printf("User was registered under ID %d\n",userID);
IEngine_FreeUser(user);
```

Related Topics

IEngine_Connect (see page 10), IEngine_RegisterUserAs (see page 66), IEngine_UpdateUser (see page 67), IEngine_RemoveUser (see page 69), User related functions (see page 11)

Advanced User Information

You may set whether fingerprint images are automatically stored in the database or not through CFG_STORE_IMAGES (see page 103) parameter. Face and iris images can't be stored in database.

2.1.5.2 IEngine_RegisterUserAs Function

Adds user to database and sets its id number

C++

```
IDKIT_API int IEngine_RegisterUserAs(const IENGINE_USER user, int userID);
```

Description

This function registers a previously initialized user in the currently open database. The user id is set to the value specified by the caller. Valid user id is a positive integer (greater or equal to 1).

Parameters

`const IENGINE_USER user`

[in] Contains user data to be added to database

`int userID`

[in] id to be attached to newly registered user

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_DBFULL	Exceeded database user limit.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_DUPLICATEID	Specified id already exists.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_Connect (see page 10), IEngine_RegisterUser (see page 65), IEngine_UpdateUser (see page 67), IEngine_RemoveUser (see page 69), User related functions (see page 11)

Advanced User Information

You may set whether fingerprint images are automatically stored in the database or not through CFG_STORE_IMAGES (see page 103) parameter. Face and iris images can't be stored in database.

2.1.5.3 IEngine_UpdateUser Function

Updates user data contained in database

C++

```
IDKIT_API int IEngine_UpdateUser(const IENGINE_USER user, int userID);
```

Description

This function updates information for a particular user already contained in the database under specified user id.

Parameters

`const IENGINE_USER user`

[in] Contains user data to be saved in database

`int userID`

[in] user id, specified id must already exist

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_Connect (see page 10), IEngine_RegisterUser (see page 65), IEngine_RegisterUserAs (see page 66), IEngine_RemoveUser (see page 69), User related functions (see page 11)

Advanced User Information

You may set whether fingerprint images are automatically stored in the database or not through CFG_STORE_IMAGES (

see page 103) parameter. Face and iris images can't be stored in database.

2.1.5.4 IEngine_UserExists Function

Checks if user already exists in the database

C++

```
IDKIT_API int IEngine_UserExists(int userID, int * userExists);
```

Description

This function checks if user already exists in the database User related functions (see page 11).

Parameters

`int userID`
[in] id of user to be loaded from database

`int * userExists`
[out] On return, If user is present in the database userExists will be 1, otherwise 0

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_DBACCESSDENIED	Database file access is denied.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_GetUser (see page 69), User related functions (see page 11)

2.1.5.5 IEngine_GetUserIfExists Function

Checks if user already exists in the database and retrieves user data from database

C++

```
IDKIT_API int IEngine_GetUserIfExists(IENGINE_USER user, int userID, int * userExists);
```

Description

This function loads all previously registered user data (biometric templates, fingerprint images, custom data) from the database into the provided user structure. User data loaded into provided user structure may subsequently be accessed through User related functions (see page 11).

Parameters

`IENGINE_USER user`
[out] On return, user data will be loaded into this structure

`int userID`
[in] id of user to be loaded from database

`int * userExists`
[out] On return, If user is present in the database userExists will be 1, otherwise 0

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.

IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_DBACCESSDENIED	Database file access is denied.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_GetUser (see page 69), User related functions (see page 11)

2.1.5.6 IEngine_GetUser Function

Retrieves user data from database

C++

```
IDKIT_API int IEngine_GetUser(IENGINE_USER user, int userID);
```

Description

This function loads all previously registered user data (biometrics templates, fingerprint images, custom data) from the database into the provided user structure. User data loaded into provided user structure may subsequently be accessed through User related functions (see page 11).

Parameters

`IENGINE_USER user`

[out] On return, user data will be loaded into this structure

`int userID`

[in] id of user to be loaded from database

Warning

Before calling this function, user structure has to be initialized through IEngine_InitUser (see page 12) function.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_BADCRYPTKEY	Invalid cipher key. Data CRC check failed after decryption.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_RegisterUser (see page 65), User related functions (see page 11)

2.1.5.7 IEngine_RemoveUser Function

Removes user from database

C++

```
IDKIT_API int IEngine_RemoveUser(int userID);
```

Description

This function flags user under specified user id as deleted in database. It flags both user templates and fingerprint images stored under specified user id. It is database maintainer responsibility to delete flagged users, tags and images.

Parameters

`int userID`

[in] user id, specified id must exist in the database

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

Function performs soft-delete, i.e. user record is not removed from the database, just delete flag is set on in the column DELETED.

Related Topics

IEngine_Connect (see page 10), IEngine_RegisterUser (see page 65), IEngine_RegisterUserAs (see page 66), IEngine_UpdateUser (see page 67), User related functions (see page 11)

2.1.5.8 IEngine_ClearDatabase Function

Deletes all entries from database

C++

```
IDKIT_API int IEngine_ClearDatabase();
```

Description

This function deletes all entries contained in database leaving an empty database at the end.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_BADCRYPTKEY	Invalid cipher key. Data CRC check failed after decryption.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

2.1.5.9 IEngine_GetUserCount Function

Returns total number of users registered in database

C++

```
IDKIT_API int IEngine_GetUserCount(int * userCount);
```

Description

This function retrieves total number of users registered in database.

Parameters

`int * userCount`

[out] On return, contains total number of registered users

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_RegisterUser (see page 65), IEngine_GetUserLimit (see page 98)

2.1.5.10 IEngine_GetAllUserIDs Function

C++

```
IDKIT_API int IEngine_GetAllUserIDs(IENGINE_COLLECTION collection);
```

Parameters

`IENGINE_COLLECTION collection`

[in] Collection that receives the list of IDs.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NULLPARAM	Parameter collection must not be NULL.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

Retrieves list of all user IDs in the database.

Related Topics

IEngine_GetUserIDsByQuery (see page 71), IENGINE_COLLECTION (see page 103)

2.1.5.11 IEngine_GetUserIDsByQuery Function

C++

```
IDKIT_API int IEngine_GetUserIDsByQuery(IENGINE_COLLECTION collection, const char * query);
```

Parameters

`IENGINE_COLLECTION collection`

[in] Collection that receives the list of IDs.

`const char * query`

[in] Tag query describing the set of IDs to retrieve.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NULLPARAM	Parameters collection and query must not be NULL.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

Retrieves list of user IDs in the database that match tag query (see page 133).

Related Topics

IEngine_GetAllUserIDs (see page 71), IENGINE_COLLECTION (see page 103)

2.1.6 Identification

Functions

	Name	Description
	IEngine_FindUser (see page 72)	Identifies user in the database.
	IEngine_FindUserInSelection (see page 73)	Identifies user in a subset of the database.
	IEngine_FindUserByQuery (see page 74)	Identifies user in a subset of the database defined by query.
	IEngine_FindUserInMemory (see page 74)	Identifies user in an external database.
	IEngine_FindFingerprint (see page 75)	Identifies particular fingerprint in the database.
	IEngine_FindFingerprintInSelection (see page 76)	Identifies particular fingerprint in a subset of the database.
	IEngine_FindFingerprintByQuery (see page 77)	Identifies particular fingerprint in a subset of the database defined by query.
	IEngine_FindFingerprintInMemory (see page 78)	Identifies particular fingerprint in an external database.

2.1.6.1 IEngine_FindUser Function

Identifies user in the database.

C++

```
IDKIT_API int IEngine_FindUser(const IENGINE_USER user, int * userID, int * score);
```

Parameters

`const IENGINE_USER user`

[in] Probe user, user to be identified in the database. The identification search is based on fingerprint templates and templates of other supported biometric modalities contained in the user structure.

`int * userID`

[out] On return, array contains user IDs of the best matching candidates.

If userID parameter is NULL on input, the parameter is ignored and not filled on return. When there are no candidates userID is 0.

`int * score`

[out] On return, array contains similarity scores of the best matching candidates.

If score parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.

IENGINE_E_NOTEMPLATES	User structure contains no templates (void user).
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function performs one-to-many identification (see page 129). It consolidates (see page 132) partial fingerprint scores and scores of other supported biometric modalities, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID and score parameters. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT.

Related Topics

IEngine_FindFingerprint (see page 75), IEngine_MatchUser (see page 86), IEngine_FindUserInSelection (see page 73), IENGINE_CONFIG (see page 103)

2.1.6.2 IEngine_FindUserInSelection Function

Identifies user in a subset of the database.

C++

```
IDKIT_API int IEngine_FindUserInSelection(const IENGINE_USER user, int selectionCount,
const int * selectedUserIDs, int * userID, int * score);
```

Parameters

`const IENGINE_USER user`

[in] Probe user, user to be identified in the database. The identification search is based on fingerprint templates and templates of other supported biometric modalities contained in the user structure.

`int selectionCount`

[in] Number of elements (users IDs) in the array selectedIDs defining the scope of the identification search

`int * userID`

[out] On return, array contains user IDs of the best matching candidates.

If userID parameter is NULL on input, the parameter is ignored and not filled on return.

`int * score`

[out] On return, array contains similarity scores of the best matching candidates.

If score parameter is NULL on input, the parameter is ignored and not filled on return.

`selectedIDs`

[in] Array of user IDs defining the scope of the identification search

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NULLPARAM	Parameters selectionCount and selectedUserIDs must not be NULL.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_NOTEMPLATES	User structure contains no templates (void user).
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADUSERID	At least one of the specified ids is not valid.

Remarks

This function performs one-to-many identification (see page 129) over selection (see page 132). It consolidates (see page 132) partial fingerprint scores and scores of other supported biometric modalities, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID and score parameters. If selectedIDs array doesn't contain any IDs, function will return userID = 0 and score = 0.

Related Topics

IEngine_FindFingerprintInSelection (see page 76), IEngine_FindUser (see page 72), IEngine_MatchUser (see page 86), IENGINE_CONFIG (see page 103)

2.1.6.3 IEngine_FindUserByQuery Function

Identifies user in a subset of the database defined by query.

C++

```
IDKIT_API int IEngine_FindUserByQuery(const IENGINE_USER user, const char * query, int *
uids, int * scores);
```

Parameters

`const IENGINE_USER user`

[in] Probe user to be identified in the database.

`const char * query`

[in] Tag query (see page 133) defining subset of the database.

`userID`

[out] On return, array contains user IDs of the best matching candidates or zeroes if there is no match. Ignored if NULL.

`score`

[out] On return, array contains similarity scores (see page 133) of the best matching candidates or zeroes if there is no match. Ignored if NULL.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NOTEMPLATES	User structure contains no templates (void user).
IENGINE_E_NULLPARAM	Parameter query must not be NULL.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function performs one-to-many identification (see page 129) over database subset defined by tag query (see page 133). It consolidates (see page 132) partial fingerprint scores and scores of other supported biometric modalities, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID and score parameters. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT.

Related Topics

IEngine_FindFingerprintByQuery (see page 77), IEngine_FindUserInSelection (see page 73), IEngine_FindUser (see page 72)

2.1.6.4 IEngine_FindUserInMemory Function

Identifies user in an external database.

C++

```
IDKIT_API int IEngine_FindUserInMemory(IENGINE_USER user, int recordCount, const unsigned
char ** recordsInMemory, int * userID, int * score);
```

Parameters

`IENGINE_USER user`

[in] Probe user, user to be identified in the database.

`int recordCount`

[in] Number of templates in array recordsInMemory.

`const unsigned char ** recordsInMemory`

[in] Array of user templates (see page 126) from external database (see page 121) to be searched through.

`int * userID`

[out] On return, array contains indices (pointing into the recordsInMemory array) of the best matching candidates or -1 if there is no match. Ignored if NULL.

`int * score`

[out] On return, array contains similarity scores (see page 133) of the best matching candidates or zeroes if there is no match. Ignored if NULL.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NOTEMPLATES	User structure contains no templates (void user).
IENGINE_E_NULLPARAM	Parameters recordCount and recordsInMemory must not be NULL.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function performs one-to-many identification (see page 129) over external database (see page 121). It consolidates (see page 132) partial fingerprint scores and scores of other supported biometric modalities, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID and score parameters. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT. This function expects that all records in external database (see page 121) (recordsInMemory) are of the latest ICS template version. In case you want to use ICS templates exported (IEngine_ExportUserTemplate (see page 56)) by any older IDKit version you should re-import them (IEngine_ImportUserTemplate (see page 57) followed by IEngine_ExportUserTemplate (see page 56)). It ensures that the template format is converted to the latest, otherwise the identification will result in undefined behavior or fatal error.

Related Topics

IEngine_FindUser (see page 72), IEngine_FindFingerprintInMemory (see page 78), IEngine_MatchUsers (see page 87), IEngine_MatchUsersModalities (see page 88)

2.1.6.5 IEngine_FindFingerprint Function

Identifies particular fingerprint in the database.

C++

```
IDKIT_API int IEngine_FindFingerprint(const IENGINE_USER user, int fingerprintIndex, int *
userID, int * bestIndex, int * score);
```

Parameters

`const IENGINE_USER user`

[in] User structure containing fingerprint to be identified in the database.

`int fingerprintIndex`

[in] Index of the fingerprint stored in the user structure. Fingerprint corresponding to this index will be identified in the database.

`int * userID`

[out] On return, array contains user IDs of the best matching candidates

If userID parameter is NULL on input, then the parameter is ignored and not filled on return.

`int * bestIndex`

[out] On return, array contains indices of the best matching fingerprints. This is useful if the identified candidate, which id=userID, contains multiple fingerprints. In this case, bestIndex permits you to determine the fingerprint with the highest similarity.

If bestIndex parameter is NULL on input, the parameter is ignored and not filled on return.

`int * score`

[out] On return, array contains similarity scores of the best matching candidates. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT.

If score parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function performs one-to-many identification (see page 129). It consolidates (see page 132) partial fingerprint scores, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID, bestIndex, and score parameters. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT.

Related Topics

IEngine_FindFingerprint, IEngine_MatchUser (see page 86), IEngine_FindFingerprintInSelection (see page 76), IENGINE_CONFIG (see page 103)

2.1.6.6 IEngine_FindFingerprintInSelection Function

Identifies particular fingerprint in a subset of the database.

C++

```
IDKIT_API int IEngine_FindFingerprintInSelection(const IENGINE_USER user, int index, int selectionCount, const int * selectedUserIDs, int * userID, int * bestIndex, int * score);
```

Parameters

`const IENGINE_USER user`

[in] User structure containing fingerprint to be identified in the database.

`int selectionCount`

[in] Number of elements (users IDs) in the array selectedIDs defining the scope of the identification search

`int * userID`

[out] On return, array contains user IDs of the best matching candidates

If userID parameter is NULL on input, then the parameter is ignored and not filled on return.

`int * bestIndex`

[out] On return, array contains indices of the best matching fingerprints. This is useful if the identified candidate, which id=userID, contains multiple fingerprints. In this case, bestIndex permits you to determine the fingerprint with the highest similarity.

If bestIndex parameter is NULL on input, the parameter is ignored and not filled on return.

`int * score`

[out] On return, array contains similarity scores of the best matching candidates.

If score parameter is NULL on input, the parameter is ignored and not filled on return.

`fingerprintIndex`

[in] Index of the fingerprint stored in the user structure. Fingerprint corresponding to this index will be identified in the database.

`selectedIDs`

[in] List of user IDs defining the scope of the identification search

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NULLPARAM	Parameters selectionCount and selectedUserIDs must not be NULL.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.

IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADUSERID	At least one of the specified ids is not valid.

Remarks

This function performs one-to-many identification (see page 129) over selection (see page 132). It consolidates (see page 132) partial fingerprint scores, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID, bestIndex, and score parameters. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT. If selectedIDs array doesn't contain any IDs, function will return userID = 0, bestIndex = 0 and score = 0.

Related Topics

IEngine_FindFingerprint (see page 75), IEngine_MatchUser (see page 86), IEngine_FindFingerprintInSelection, IENGINE_CONFIG (see page 103)

2.1.6.7 IEngine_FindFingerprintByQuery Function

Identifies particular fingerprint in a subset of the database defined by query.

C++

```
IDKIT_API int IEngine_FindFingerprintByQuery(const IENGINE_USER user, int index, const char * query, int * uids, int * bestIndices, int * scores);
```

Parameters

`const IENGINE_USER user`

[in] User structure containing fingerprint to be identified in the database.

`int index`

[in] Index of the fingerprint stored in the user structure. Fingerprint corresponding to this index will be identified in the database.

`const char * query`

[in] Tag query (see page 133) defining subset of the database.

`int * uids`

[out] On return, array contains user IDs of the best matching candidates or zeroes if there is no match. Ignored if NULL.

`int * bestIndices`

[out] On return, array contains the index of the best matching fingerprints for each matching user or zeroes if there is no match. Ignored if NULL.

`int * scores`

[out] On return, array contains similarity scores (see page 133) of the best matching candidates or zeroes if there is no match. Ignored if NULL.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NULLPARAM	Parameter query must not be NULL.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function performs one-to-many identification (see page 129) over database subset defined by tag query (see page 133). It consolidates (see page 132) partial fingerprint scores, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID, bestIndex, and score parameters. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT.

Related Topics

IEngine_FindFingerprint (see page 75), IEngine_FindFingerprintInSelection (see page 76), IEngine_FindUserByQuery (see page 74)

2.1.6.8 IEngine_FindFingerprintInMemory Function

Identifies particular fingerprint in an external database.

C++

```
IDKIT_API int IEngine_FindFingerprintInMemory(const IENGINE_USER user, int index, int recordCount, const unsigned char ** recordsInMemory, int * userID, int * bestIndex, int * score);
```

Parameters

`const IENGINE_USER user`

[in] User structure containing fingerprint to be identified in the database.

`int index`

[in] Index of a fingerprint in the user structure. Fingerprint at this index will be identified in the database.

`int recordCount`

[in] Number of templates in array recordsInMemory.

`const unsigned char ** recordsInMemory`

[in] Array of user templates (see page 126) from external database (see page 121) to be searched through.

`int * userID`

[out] On return, array contains indices (pointing into the recordsInMemory array) of the best matching candidates or -1 if there is no match. Ignored if NULL.

`int * bestIndex`

[out] On return, array contains the index of the best matching fingerprint for each matching user or zeroes if there is no match. Ignored if NULL.

`int * score`

[out] On return, array contains similarity scores (see page 133) of the best matching candidates or zeroes if there is no match. Ignored if NULL.

Returns

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero. For a list of error codes, see section Error Codes (see page 115).

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NULLPARAM	Parameters recordCount and recordsInMemory must not be NULL.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function performs one-to-many identification (see page 129) over external database (see page 121). It consolidates (see page 132) partial fingerprint scores, filters out candidates below CFG_SIMILARITY_THRESHOLD, and saves the best results in userID, bestIndex, and score parameters. Size of these arrays must be equal to the value of CFG_BEST_CANDIDATES_COUNT. This function expects that all records in external database (see page 121) (recordsInMemory) are of the latest ICS template version. In case you want to use ICS templates exported (IEngine_ExportUserTemplate (see page 56)) by any older IDKit version you should re-import them (IEngine_ImportUserTemplate (see page 57) followed by IEngine_ExportUserTemplate (see page 56)). It ensures that the template format is converted to the latest, otherwise the identification will result in undefined behavior or fatal error.

Related Topics

IEngine_FindFingerprint (see page 75), IEngine_FindUserInMemory (see page 74), IEngine_MatchFingerprints (see page 80)

2.1.7 Verification

Functions

	Name	Description
	IEngine_MatchFingerprint (see page 79)	Compares a particular fingerprint with fingerprints of another user
	IEngine_MatchFingerprints (see page 80)	Compares two fingerprints
	IEngine_MatchFingerprints_transformation (see page 81)	Compares two fingerprints and returns corresponding transformation.
	IEngine_MatchFace (see page 82)	Compares a particular face template with face templates of another user registered in the database.
	IEngine_MatchFaces (see page 83)	Compares two face templates.
	IEngine_MatchIris (see page 84)	Compares a particular iris template with iris templates of another user registered in the database.
	IEngine_MatchIrises (see page 85)	Compares two iris templates.
	IEngine_MatchUser (see page 86)	Compares supported biometric modalities of the input user with biometric modalities of a particular user registered in the database.
	IEngine_MatchUsers (see page 87)	Compares two users based on supported biometric modalities contained in user structure.
	IEngine_MatchUsersModalities (see page 88)	Compares two users based on desired supported biometric modalities.

2.1.7.1 IEngine_MatchFingerprint Function

Compares a particular fingerprint with fingerprints of another user

C++

```
IDKIT_API int IEngine_MatchFingerprint(const IENGINE_USER user, int fingerprintIndex, int userID, int * bestIndex, int * score);
```

Description

This function compares a particular fingerprint belonging to probe user against all corresponding fingerprints belonging to reference user. The fingerprint with index equal to the fingerprintIndex parameter is compared against each corresponding fingerprint belonging to reference user.

Two fingerprints are considered as corresponding fingerprints if they have the same finger position (IENGINE_FINGER_POSITION (see page 107)), or if at least one of the fingerprints has a position indexed as UNKNOWN_POSITION.

If the reference user structure contains more than one corresponding fingerprint, the resulting similarity score is a consolidation of partial similarity scores.

Only scores above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103) are returned. If the similarity score is not above this threshold, score parameter will be set to 0.

Parameters

`const IENGINE_USER user`

[in] Probe user, user whose fingerprint will be compared with reference user fingerprints.

`int fingerprintIndex`

[in] index of probe fingerprint that will be compared against the fingerprints of reference user

`int userID`

[in] ID of reference user under which the reference user is registered in the database.

`int * bestIndex`

[out] On return, contains the index of the best matching fingerprint. This is useful if reference user contains multiple fingerprints. In this case, bestIndex permits you to determine the fingerprint with the highest similarity.

If bestIndex parameter is NULL on input, the parameter is ignored and not filled on return.

`int * score`

[out] On return, contains the similarity score between the matched fingerprint and the corresponding fingerprint (or possibly the set of corresponding

fingerprints from the reference user structure.

If score parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADUSERID	Specified id is not valid.

Related Topics

IEngine_FindFingerprint (see page 75), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_MAX_ROTATION (see page 103) defines maximum allowed rotation between two fingerprint images.
- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the current similarity score does not reach this threshold, score parameter is set to 0.

2.1.7.2 IEngine_MatchFingerprints Function

Compares two fingerprints

C++

```
IDKIT_API int IEngine_MatchFingerprints(const IENGINE_USER probeUser, int fingerprintIndex,
const IENGINE_USER galleryUser, int fingerprintIndex2, int * score);
```

Description

This function compares a particular fingerprint belonging to probe user against a fingerprint belonging to gallery user.

The resulting score will have a non-zero value only if it is above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103). If the similarity score is below this threshold, the score parameter will be set to 0.

Parameters

`const IENGINE_USER probeUser`
[in] Probe user containing the probe fingerprint

`int fingerprintIndex`
[in] Index of the probe fingerprint that will be compared with the gallery fingerprint

`const IENGINE_USER galleryUser`
[in] Gallery user containing the gallery fingerprint

`int fingerprintIndex2`
[in] index of the gallery fingerprint that will be compared with the probe fingerprint

`int * score`
[out] On return, contains the similarity score between the two compared fingerprints

If score parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.

IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NOFINGERPRINT	User structure contains no fingerprint. This operation is not commutative, different order of users will lead to a different matching score. For better understanding, the probe user
(probeUser	first parameter of function) determines how the users will be matched. Changing the probe user causes the users will be matched slightly different way.

Related Topics

IEngine_MatchUser (see page 86), IEngine_MatchUsers (see page 87), IEngine_MatchUsersModalities (see page 88), IEngine_MatchFingerprints, IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_MAX_ROTATION (see page 103) defines maximum allowed rotation between two fingerprint images.
- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the current similarity score does not reach this threshold, score parameter is set to 0.

2.1.7.3 IEngine_MatchFingerprints_transformation Function

Compares two fingerprints and returns corresponding transformation.

C++

```
IDKIT_API int IEngine_MatchFingerprints_transformation(const IENGINE_USER probeUser, int
index, const IENGINE_USER galleryUser, int index2, int * score, int * dx, int * dy,
unsigned char * dAngle);
```

Description

This function compares a particular fingerprint belonging to probe user against a fingerprint belonging to gallery user.

The resultig score will have a non-zero value only if it is above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103). If the similarity score is below this threshold, the score parameter will be set to 0.

Returned transformation defines translation and rotation between the two fingerprints. The parameters can be used to calculate corresponding minutiae points positions using these equations: a = dAngle*PI/128 x2 = x*cos(a)-y*sin(a)+dx y2 = x*sin(a)+y*cos(a)+dy

This operation is not commutative, different order of users will lead to a different matching score. For better understanding, the probe user (probeUser - first parameter of function) determines how the users will be matched. Changing the probe user causes the users will be matched slightly different way.

Parameters

`const IENGINE_USER probeUser`
[in] Probe user containing the probe fingerprint

`const IENGINE_USER galleryUser`
[in] Gallery user containing the gallery fingerprint

`int * score`
[out] On return, contains the similarity score between the two compared fingerprints

`int * dx`
[out] On return, contains x-axis deviation

`int * dy`
[out] On return, contains y-axis deviation

`unsigned char * dAngle`
[out] On return, contains angle deviation in degrees

If score parameter is NULL on input, the parameter is ignored and not filled on return. If dx, dy, dAngle are NULL on input, the parameters are ignored and not

filled on return.

`fingerprintIndex`

[in] Index of the probe fingerprint that will be compared with the gallery fingerprint

`fingerprintIndex2`

[in] index of the gallery fingerprint that will be compared with the probe fingerprint

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_NOFINGERPRINT	User structure contains no fingerprint.

Related Topics

IEngine_MatchFingerprints (see page 80), IEngine_MatchUser (see page 86), IEngine_MatchUsers (see page 87), IEngine_MatchUsersModalities (see page 88), IEngine_MatchFingerprints (see page 80), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_MAX_ROTATION (see page 103) defines maximum allowed rotation between two fingerprint images.
- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the current similarity score does not reach this threshold, score parameter is set to 0.

2.1.7.4 IEngine_MatchFace Function

Compares a particular face template with face templates of another user registered in the database.

C++

```
IDKIT_API int IEngine_MatchFace(const IENGINE_USER user, int index, int uid, int *
bestIndex, int * score);
```

Description

This function compares face template with particular index belonging to probe user against all face templates belonging to reference user. If the reference user structure contains more than one face templates, the resulting similarity score is a consolidation of partial similarity scores. Only resulting scores above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103) are returned. If the similarity score is not above this threshold, score parameter will be set to 0.

Parameters

`const IENGINE_USER user`

[in] Probe user, user whose face template will be compared with reference user face templates

`int index`

[in] Index of probe face template that will be compared against the face templates of reference user

`int * bestIndex`

[out] On return, contains the index of the best matching face template from registered user. This is useful if reference user contains multiple face templates. In this case, bestIndex permits you to determine the face template with the highest similarity score. If bestIndex parameter is NULL on input, the parameter is ignored and not filled on return.

`int * score`

[out] On return, contains the similarity score between the probe face template and all face templates from the reference user structure (registered user). If score parameter is NULL on input, the parameter is ignored and not filled on return.

`userID`

[in] ID of reference user under which the reference user is registered in the database.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_NOFACES	User structure contains no face templates.
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_MatchFaces (see page 83), IEngine_MatchFingerprint (see page 79), IEngine_FindFingerprint (see page 75), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the current similarity score does not reach this threshold, score parameter is set to 0.

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.7.5 IEngine_MatchFaces Function

Compares two face templates.

C++

```
IDKIT_API int IEngine_MatchFaces(const IENGINE_USER user, int index, const IENGINE_USER user2, int index2, int * score);
```

Description

This function compares a particular face template belonging to probe user against a face template belonging to reference user.

The resultig score will have a non-zero value only if it is above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103). If the similarity score is below this threshold, the score parameter will be set to 0.

Parameters

`const IENGINE_USER user`
[in] Probe user

`int index`
[in] Index of the probe face template that will be compared with the reference face template

`const IENGINE_USER user2`
[in] Reference user

`int index2`
[in] index of the reference face template that will be compared with the probe face template

`int * score`
[out] On return, contains the similarity score between the two compared face templates

If score parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NOFACES	User structure contains no face templates.
IENGINE_E_NOTCONNECTED	IDKit not connected.

Related Topics

IEngine_MatchFace (see page 82), IEngine_MatchUser (see page 86), IEngine_MatchUsers (see page 87), IEngine_MatchUsersModalities (see page 88), IEngine_MatchFingerprints (see page 80), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the current similarity score does not reach this threshold, score parameter is set to 0.

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.7.6 IEngine_MatchIris Function

Compares a particular iris template with iris templates of another user registered in the database.

C++

```
IDKIT_API int IEngine_MatchIris(const IENGINE_USER user, int index, int uid, int *
bestIndex, int * score);
```

Description

This function compares iris template with particular index belonging to probe user against all iris templates belonging to reference user. If the reference user structure contains more than one iris templates, the resulting similarity score is a consolidation of partial similarity scores. Only resulting scores above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103) are returned. If the similarity score is not above this threshold, score parameter will be set to 0.

Parameters

`const IENGINE_USER user`

[in] Probe user, user whose iris template will be compared with reference user iris templates

`int index`

[in] Index of probe iris template that will be compared against the iris templates of reference user

`int * bestIndex`

[out] On return, contains the index of the best matching iris template from registered user. This is useful if reference user contains multiple iris templates. In this case, bestIndex permits you to determine the iris template with the highest similarity score. If bestIndex parameter is NULL on input, the parameter is ignored and not filled on return.

`int * score`

[out] On return, contains the similarity score between the probe iris template and all iris templates from the reference user structure (registered user). If score parameter is NULL on input, the parameter is ignored and not filled on return.

`userID`

[in] ID of reference user under which the reference user is registered in the database.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.

IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NOIRIS	User structure contains no iris templates.

Related Topics

IEngine_MatchIrises (see page 85), IEngine_MatchFingerprint (see page 79), IEngine_FindFingerprint (see page 75), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the current similarity score does not reach this threshold, score parameter is set to 0.

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.7.7 IEngine_MatchIrises Function

Compares two iris templates.

C++

```
IDKIT_API int IEngine_MatchIrises(const IENGINE_USER user, int index, const IENGINE_USER user2, int index2, int * score);
```

Description

This function compares a particular iris template belonging to probe user against a iris template belonging to reference user.

The resultig score will have a non-zero value only if it is above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103). If the similarity score is below this threshold, the score parameter will be set to 0.

Parameters

`const IENGINE_USER user`

[in] Probe user

`int index`

[in] Index of the probe iris template that will be compared with the reference iris template

`const IENGINE_USER user2`

[in] Reference user

`int index2`

[in] index of the reference iris template that will be compared with the probe iris template

`int * score`

[out] On return, contains the similarity score between the two compared iris templates

If score parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_NOIRIS	User structure contains no iris templates.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NOTCONNECTED	IDKit not connected.

Related Topics

IEngine_MatchIris (see page 84), IEngine_MatchUser (see page 86), IEngine_MatchUsers (see page 87), IEngine_MatchUsersModalities (see page 88), IEngine_MatchFingerprints (see page 80), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the current similarity score does not reach this threshold, score parameter is set to 0.

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.7.8 IEngine_MatchUser Function

Compares supported biometric modalities of the input user with biometric modalities of a particular user registered in the database.

C++

```
IDKIT_API int IEngine_MatchUser(IENGINE_USER user, int userID, int * score, int *
matchingFingersCount);
```

Description

This function compares two users - probe user and reference user. The comparison (1-to-1 matching) is based on supported biometric modalities contained in each user structure. Each probe modality (more precisely, each modality template belonging to probe user) is compared against each corresponding modality belonging to the reference user.

Two fingerprints are considered as corresponding fingerprints if they have the same finger position (IENGINE_FINGER_POSITION (see page 107)), or if at least one of the fingerprints has a position indexed as UNKNOWN_POSITION.

If the probe and the reference user structures contain more than one corresponding fingerprint pair, the resulting similarity score is a consolidation of partial similarity scores. Similarly if the probe and the reference user structures contain more than one template of other supported biometric modalities, the resulting similarity score is a consolidation of partial similarity scores.

If the probe and the reference user structures contain more than one kind of modality, the resulting similarity score is a consolidation of partial similarity scores.

Only scores above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103) are returned. If the similarity score is not above this threshold, score parameter will be set to 0.

Parameters

`IENGINE_USER user`

[in] Probe user, user to be compared with reference user.

`int userID`

[in] ID of reference user under which the reference user is registered in the database

`int * score`

[out] On return, array contains the similarity score between two compared users. If score parameter is NULL on input, the parameter is ignored and not filled on return.

`int * matchingFingersCount`

[out] On returns , contains number of matching fingers, If the parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.

IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADUSERID	Specified id is not valid.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_NOTEMPLATES	User structure contains no templates (void user).
IENGINE_E_NONEXISTINGID	Specified id not found in database.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_MatchUsers (see page 87), IEngine_MatchUsersModalities (see page 88), IEngine_MatchFingerprints (see page 80), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_MAX_ROTATION (see page 103) defines maximum allowed rotation between two fingerprint images
- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the similarity score between compared users does not reach this threshold, score parameter is set to 0.

2.1.7.9 IEngine_MatchUsers Function

Compares two users based on supported biometric modalities contained in user structure.

C++

```
IDKIT_API int IEngine_MatchUsers(const IENGINE_USER probeUser, const IENGINE_USER
galleryUser, int * score, int * matchingFingersCount);
```

Description

This function compares two users - probe user and gallery user. The comparison (1-to-1 matching) is based on supported biometric modalities contained in each user structure. Each probe modality (more precisely, each modality template belonging to probe user) is compared against each corresponding modality belonging to the gallery user.

Two fingerprints are considered as corresponding fingerprints if they have the same finger position (IENGINE_FINGER_POSITION (see page 107)), or if at least one of the fingerprints has a position indexed as UNKNOWN_POSITION.

This operation is not commutative, different order of users will lead to a different matching score. For better understanding, the probe user (probeUser - first parameter of function) determines how the users will be matched. Changing the probe user causes the users will be matched slightly different way.

If the probe and the gallery user structures contain more than one corresponding fingerprint pair, the resulting similarity score is a consolidation of partial similarity scores. Similarly if the probe and the gallery user structures contain more than one template of other supported biometric modalities, the resulting similarity score is a consolidation of partial similarity scores.

If the probe and the gallery user structures contain more than one kind of modality, the resulting similarity score is a consolidation of partial similarity scores.

Only scores above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103) are returned. If the similarity score is not above this threshold, score parameter will be set to 0.

Parameters

`const IENGINE_USER probeUser`

[in] Probe user, user to be compared with gallery user.

`const IENGINE_USER galleryUser`

[in] Gallery user

`int * score`

[out] On return, contains the similarity score between two compared users.

If score parameter is NULL on input, the parameter is ignored and not filled on return.

`int * matchingFingersCount`

[out] On return, contains number of matching fingers If the parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NOTEMPLATES	At least one of the input user structures contains no templates (void user).
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_MatchUser (see page 86), IEngine_MatchUsersModalities (see page 88), IEngine_MatchFingerprints (see page 80), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_MAX_ROTATION (see page 103) defines maximum allowed rotation between two fingerprint images
- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the similarity score between compared users does not reach this threshold, score parameter is set to 0.

2.1.7.10 IEngine_MatchUsersModalities Function

Compares two users based on desired supported biometric modalities.

C++

```
IDKIT_API int IEngine_MatchUsersModalities(const IENGINE_USER user, const IENGINE_USER user2, int modalities, int * score);
```

Description

This function compares two users - probe user and reference user. The comparison (1-to-1 matching) is based on desired supported biometric modalities contained in each user structure. Each probe desired modality (more precisely, each modality template belonging to probe user) is compared against each corresponding modality belonging to the reference user.

Two fingerprints are considered as corresponding fingerprints if they have the same finger position (IENGINE_FINGER_POSITION (see page 107)), or if at least one of the fingerprints has a position indexed as UNKNOWN_POSITION.

If the probe and the reference user structures contain more than one corresponding fingerprint pair, the resulting similarity score is a consolidation of partial similarity scores. Similarly if the probe and the reference user structures contain more than one template of other supported biometric modalities, the resulting similarity score is a consolidation of partial similarity scores.

If the probe and the reference user structures contain more than one kind of modality, the resulting similarity score is a consolidation of partial similarity scores.

Only scores above the threshold defined by parameter CFG_SIMILARITY_THRESHOLD (see page 103) are returned. If the similarity score is not above this threshold, score parameter will be set to 0.

Parameters

`const IENGINE_USER user`

[in] Probe user, user to be compared with reference user.

`const IENGINE_USER user2`

[in] Reference user.

`int modalities`

[in] Modalities according to (IENGINE_MATCHING_MODALITY (see page 112))

`int * score`

[out] On return, contains the similarity score between two compared users. If score parameter is NULL on input, the parameter is ignored and not filled on return.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_NOTEMPLATES	At least one of the input user structures contains no templates (void user).
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADUSERID	Specified id is not valid.

Related Topics

IEngine_MatchUser (see page 86), IEngine_MatchUsers (see page 87), IEngine_MatchFingerprints (see page 80), IENGINE_CONFIG (see page 103)

Advanced User Information

The behaviour of this function may be controlled through following parameters:

- CFG_MAX_ROTATION (see page 103) defines maximum allowed rotation between two fingerprint images
- CFG_SIMILARITY_THRESHOLD (see page 103) defines minimum acceptable similarity score.

If the similarity score between compared users does not reach this threshold, score parameter is set to 0.

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.8 Images

Functions

	Name	Description
	IEngine_ConvertRawToImage (see page 89)	Converts raw image into chosen image format.
	IEngine_ConvertImageToRaw (see page 90)	Converts an image into raw image.
	IEngine_ConvertImage (see page 91)	Converts an image into another image format

2.1.8.1 IEngine_ConvertRawToImage Function

Converts raw image into chosen image format.

C++

```
IDKIT_API int IEngine_ConvertRawToImage(const unsigned char * rawImage, int width, int
height, IENGINE_IMAGE_FORMAT format, unsigned char * image, int * length);
```

Parameters

`const unsigned char * rawImage`
[in] Pointer to the uncompressed raw image (1 byte per pixel in grayscale)

`int width`
[in] The number of pixels indicating the width of the raw image

`int height`
[in] The number of pixels indicating the height of the raw image

`IENGINE_IMAGE_FORMAT format`
[in] Format of the output image

`unsigned char * image`
[out] Pointer to the memory space where image will be saved

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the bmpImage parameter. On return, this parameter will be equal to the total length of BMP image.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	Parameters length and rawImage must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_BADIMAGE	Invalid image or unsupported format.

Remarks

This function converts raw image into another image in one of supported image formats (Image Processing (see page 6)). The image must be of no-header uncompressed raw 8-bit (one byte per pixel) grayscale format. The beginning is the upper left corner of the image. Value of black is 0x0, white is 0xFF. This function uses two-phase buffer retrieval (see page 118) algorithm.

Example

```
...
int width,height;
unsigned char *rawImage,*image;
... //rawImage is initialized here
int length;
IEngine_ConvertRawToImage(rawImage, width, height, format, NULL, &length);
image=new unsigned char[length];
int err=IEngine_ConvertRawToImage(rawImage, width, height, format, image, &length);
if(err!=0)
    printf("Raw image was converted\n");
else
    printf("Error occurred\n");
...
delete[] image;
```

Related Topics

IEngine_ConvertImageToRaw (see page 90), IEngine_AddFingerprint (see page 14), IEngine_SetFingerprint (see page 18)

2.1.8.2 IEngine_ConvertImageToRaw Function

Converts an image into raw image.

C++

```
IDKIT_API int IEngine_ConvertImageToRaw(const unsigned char * image, int imageSize,
unsigned char * rawImage, int * width, int * height);
```

Parameters

`const unsigned char * image`

[in] Pointer to the input image

`int imageSize`

[in] Size of the image in bytes.

`unsigned char * rawImage`

[out] Pointer to the memory space where raw image (1 byte per pixel in grayscale) will be written

`int * width`

[out] The number of pixels indicating the width of the raw image

`int * height`

[out] The number of pixels indicating the height of the raw image

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_NULLPARAM	Parameters width and height must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function converts image (BMP, PNG, JPEG, GIF, TIFF, and WSQ) into uncompressed raw 8-bit (one byte per pixel) grayscale format. This function uses two-phase buffer retrieval (see page 118) algorithm with the exception that width and height is returned instead of length. Length of the raw image is width * height.

Related Topics

IEngine_ConvertRawToImage (see page 89), IEngine_GetFingerprintImage (see page 43)

2.1.8.3 IEngine_ConvertImage Function

Converts an image into another image format

C++

```
IDKIT_API int IEngine_ConvertImage(const unsigned char * inImage, int inLength,
IENGINE_IMAGE_FORMAT format, unsigned char * outImage, int * outLength);
```

Parameters

`const unsigned char * inImage`

[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)).

`IENGINE_IMAGE_FORMAT format`

[in] Format of the output image

`unsigned char * outImage`

[out] Pointer to the memory space where output image will be written

`int * outLength`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the outImage parameter. On return, this parameter will be equal to the total image length.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_NULLPARAM	Parameters inImage and outLength must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function converts image in any of supported formats (Image Processing (see page 6)) into another image format. This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_ConvertRawToImage (see page 89), IEngine_GetFingerprintImage (see page 43)

2.1.9 IEngine_GetFaceTemplate Function

Exports face template at specified index from user structure.

C++

```
IDKIT_API int IEngine_GetFaceTemplate(const IENGINE_USER user, int faceIndex, unsigned char * templateData, int * length);
```

Description

Note: Face template stored in output memory buffer is face template extracted using Innovatrics IFace SDK.

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`int faceIndex`

[in] Index of particular face template

`unsigned char * templateData`

[out] Pointer to the memory space where face template raw data will be saved

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by templateData. On return, this parameter will be equal to the total length of user template.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified face index is not valid.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_NOFACES	User structure contains no face templates.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_AddFaceTemplate (see page 31), IEngine_SetFaceTemplate (see page 32), IEngine_ExportUserTemplate (see page 56), IEngine_ImportUserTemplate (see page 57)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.10 Settings and Constants

Functions

	Name	Description
	IEngine_SetParameter (see page 93)	Sets parameter value
	IEngine_GetParameter (see page 93)	Gets parameter value
	IEngine_SetStringParameter (see page 94)	Sets string parameter value
	IEngine_GetStringParameter (see page 94)	Gets string parameter value
	IEngine_SetCryptKey (see page 95)	Sets the cipher key used for the encryption of database content

	IEngine_SetLogFile (see page 95)	Enables logging into the specified file.
	IEngine_GetProductString (see page 96)	Returns product name and version formated as string (e.g., IDKit SDK 2.25.3)
	IEngine_GetHardwareId (see page 96)	Returns hardware ID of device for licensing purposes.
	IEngine_GetLicenseInformation (see page 96)	Returns information about actually used license.
	IEngine_GetErrorMsg (see page 97)	Returns error message formated as string
	IEngine_GetMemoryUsage (see page 97)	Returns memory usage for template cache.
	IEngine_GetUserLimit (see page 98)	Returns maximum allowed number of users
	IEngine_GetHardwareIdByMethod (see page 98)	Returns hardware ID of device for licensing purposes according to the selected method.

2.1.10.1 IEngine_SetParameter Function

Sets parameter value

C++

```
IDKIT_API int IEngine_SetParameter(IENGINE_CONFIG parameter, int value);
```

Description

This function sets the value of one of the library parameters. Library parameters control the behaviour of some functions. Global parameters apply to all connections, non-global parameters are unique to a particular connection. See IENGINE_CONFIG (see page 103) for more details about parameters.

Parameters

`IENGINE_CONFIG parameter`

[in] Parameter type

`int value`

[in] New value for the parameter

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADPARAM	Invalid parameter type provided.
IENGINE_E_BADVALUE	Invalid value provided.

Related Topics

IENGINE_CONFIG (see page 103)

2.1.10.2 IEngine_GetParameter Function

Gets parameter value

C++

```
IDKIT_API int IEngine_GetParameter(IENGINE_CONFIG parameter, int * value);
```

Description

This function gets the value of one of the library parameters. Library parameters control the behaviour of some functions. See IENGINE_CONFIG (see page 103) for more details about parameters.

Parameters

`IENGINE_CONFIG parameter`

[in] Parameter type

`int * value`

[out] On return, contains value of the specified parameter

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADPARAM	Invalid parameter type provided.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.

Related Topics

IENGINE_CONFIG (see page 103)

2.1.10.3 IEngine_SetStringParameter Function

Sets string parameter value

C++

```
IDKIT_API int IEngine_SetStringParameter(IENGINE_CONFIG parameter, const char * value);
```

Description

This function sets the value of one of the library string parameters. Library parameters control the behaviour of some functions. Global parameters apply to all connections, non-global parameters are unique to a particular connection. See IENGINE_CONFIG (see page 103) for more details about parameters.

Parameters

`IENGINE_CONFIG parameter`
[in] Parameter type

`const char * value`
[in] New value for the parameter

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_BADPARAM	Invalid parameter type provided.
IENGINE_E_BADVALUE	Invalid value provided.

Related Topics

IENGINE_CONFIG (see page 103)

2.1.10.4 IEngine_GetStringParameter Function

Gets string parameter value

C++

```
IDKIT_API int IEngine_GetStringParameter(IENGINE_CONFIG parameter, char * value, int *
length);
```

Description

This function gets the value of one of the library string parameters. Library parameters control the behaviour of some functions. See IENGINE_CONFIG (see page 103) for more details about parameters.

Parameters

`IENGINE_CONFIG parameter`
[in] Parameter type

`char * value`
[out] On return, contains value of the specified parameter

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADPARAM	Invalid parameter type provided.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.

Related Topics

IENGINE_CONFIG (see page 103)

2.1.10.5 IEngine_SetCryptKey Function

Sets the cipher key used for the encryption of database content

C++

```
IDKIT_API int IEngine_SetCryptKey(const unsigned char cryptKey[CRYPT_KEY_LENGTH]);
```

Description

This function sets the cryptography cipher key used for the encryption of biometric templates and images stored in the database. The algorithm used for encryption is AES (Advanced Enrcyption Standard) cipher, with a key-size of 256-bits. If the encryption is switched on, each time when biometric templates and images are stored in the database, an encryption is automatically performed. When templates or images are loaded back from the database, they are automatically decrypted using the currently selected cipher key.

If a NULL pointer is provided on input, no encryption will be used. (Default state)

This function does nothing if you are connected to ExpressID AFIS.

Parameters

`const unsigned char cryptKey[CRYPT_KEY_LENGTH]`

[in] Specifies the cipher key, this key can be any random sequence of 32 characters

Warning

The encryption applies only for active connection so it is necessary to call this function after each IEngine_SelectConnection (see page 10) or before IEngine_Connect (see page 10). In fact, you should not change the cipher key after you made a connection to a database. All templates and images stored in the same database has to be encrypted using the same cipher key, if not data inconsistencies will occur.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

2.1.10.6 IEngine_SetLogFile Function

Enables logging into the specified file.

C++

```
IDKIT_API int IEngine_SetLogFile(const char * filename);
```

Parameters

`const char * filename`

[in] Name of the log file to open or NULL to log to console.

Returns

On success, this function returns IENGINE_E_NOERROR.

On failure, this function returns one of the errors listed in error code table (see page 115).

Remarks

Logging is enabled by setting parameter (see page 93) CFG_LOG_LEVEL (see page 103). By default, all messages go to console. This function allows selection of the file that will receive all log messages from IDKit SDK. If the file already exists, it is appended to. If the parameter is NULL or an empty string, logs are redirected back to console. File name may contain UTF8 characters.

2.1.10.7 IEngine_GetProductString Function

Returns product name and version formated as string (e.g., IDKit SDK 2.25.3)

C++

```
IDKIT_API const char * IEngine_GetProductString();
```

2.1.10.8 IEngine_GetHardwareId Function

Returns hardware ID of device for licensing purposes.

C++

```
IDKIT_API int IEngine_GetHardwareId(char * hwId, int * length);
```

Parameters

`char * hwId`

[out] Pointer to the memory space where hardware ID will be saved

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the hwId parameter. On return, this parameter will be equal to the total hardware ID length.

Returns

On success, this function returns IENGINE_E_NOERROR.

On failure, this function returns one of the errors listed in error code table (see page 115).

Remarks

This function uses two-phase buffer retrieval (see page 118) algorithm.

2.1.10.9 IEngine_GetLicenseInformation Function

Returns information about actually used license.

C++

```
IDKIT_API int IEngine_GetLicenseInformation(char * hwId, int * hwId_length, char *
clientId, int * clientId_length, int * userLimit, int * clientLimit, int * expYear, int *
expMonth, int * expDay, char * licFileLocation);
```

Parameters

`char * hwId`

[out] Pointer to the memory space where hardware ID will be saved (Note: hardware ID stored in license)

`int * hwId_length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the hwId parameter. On return, this parameter will be equal to the total hardware ID length.

`char * clientId`

[out] Pointer to the memory space where client ID will be saved

`int * clientId_length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the clientId parameter. On return, this parameter will be equal to the total client ID length.

`int * userLimit`

[out] On return, contains maximum allowed number of users in the database.

`int * clientLimit`

[out] On return, contains maximum allowed number of concurrent clients connected to ExpressID AFIS server

`int * expYear`

[out] On return, contains expiration year of the license

`int * expMonth`

[out] On return, contains expiration month of the license

`int * expDay`

[out] On return, contains expiration day of the license

`char * licFileLocation`

[out] Pointer to the memory space where path to license file will be saved (Note: maximum 4096 bytes long)

Returns

On success, this function returns IENGINE_E_NOERROR.

On failure, this function returns one of the errors listed in error code table (see page 115).

Remarks

This function uses two-phase buffer retrieval (see page 118) algorithm.

2.1.10.10 IEngine_GetErrorMsg Function

Returns error message formated as string

C++

```
IDKIT_API const char * IEngine_GetErrorMsg(int errcode);
```

Description

This function returns an error message string corresponding to the specified error code.

Parameters

`int errcode`

[in] Error code to be translated into error message

Related Topics

Error Codes (see page 115)

2.1.10.11 IEngine_GetMemoryUsage Function

Returns memory usage for template cache.

C++

```
IDKIT_API int IEngine_GetMemoryUsage(int * memoryUsage);
```

Description

This function calculates total amount of memory used by IDKit for user template cache. It does not work if the library is connected to ExpressID AFIS. For memory database the calculation includes user templates, images, custom data, tags and tag cache. For sqlite database the calculation includes user templates and tag cache (images, custom data and tags are stored only in the database).

Parameters

`int * memoryUsage`

[out] On return, contains memory usage in kB (1kB = 1024B)

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_DBOPEN	Could not connect to database.
IENGINE_E_DBFAILED	Unexpected database failure occurred.
IENGINE_E_NOTSUPPORTED	This API call is not supported for this connection type.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IEngine_RegisterUser (see page 65), IEngine_GetUserLimit (see page 98)

2.1.10.12 IEngine_GetUserLimit Function

Returns maximum allowed number of users

C++

```
IDKIT_API int IEngine_GetUserLimit(int * userLimit);
```

Description

This function retrieves maximum allowed number of users that can be registered in database. Your user limit is determined by the license you purchased.

Parameters

`int * userLimit`

[out] On return, contains maximum allowed number of users in the database

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.

Related Topics

IEngine_GetUserCount (see page 70)

2.1.10.13 IEngine_GetHardwareIdByMethod Function

Returns hardware ID of device for licensing purposes according to the selected method.

C++

```
IDKIT_API int IEngine_GetHardwareIdByMethod(IENGINE_HARDWARE_ID_METHOD method, char * hwId,
int * length);
```

Parameters

`IENGINE_HARDWARE_ID_METHOD method`

[in] Method which is going to be used for generating a hardware ID.

`char * hwId`

[out] Pointer to the memory space where hardware ID will be saved

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by the hwId parameter. On return, this parameter will be equal to the total hardware ID length.

Returns

On success, this function returns IENGINE_E_NOERROR.

On failure, this function returns one of the errors listed in error code table (see page 115).

Remarks

This function uses two-phase buffer retrieval (see page 118) algorithm.

2.1.11 IEngine_GetFingerprintClass Function

Analyzes fingerprint image and returns its class according to IAFIS specification (https://identifyus.org/help/FPinformation.pdf).

C++

```
IDKIT_API int IEngine_GetFingerprintClass(const unsigned char * fingerprintImage, int
length, int * fingerprintClass);
```

Parameters

`const unsigned char * fingerprintImage`

[in] Pointer to an image stored in memory. Image format must be one of supported image formats (Image Processing (see page 6)). Minimal allowed size of image is 90x90 pixels and maximal size of image is 4000x4000 pixel

`int length`

[in] length of fingerprintImage buffer

`int * fingerprintClass`

[out] detected fingerprint class (see IENGINE_FINGERPRINT_CLASS (see page 110))

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADIMAGE	Invalid image or unsupported image format.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_NULLPARAM	Parameter fingerprintImage must not be NULL.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Related Topics

IENGINE_FINGERPRINT_CLASS (see page 110)

Availability

This method is not available in , IDKit Multi Android SDK and IDKit Embedded SDK.

2.1.12 IEngine_GetFingerprintResizeRatio Function

Unsupported function for internal purposes.

Fingerprint image can be rescaled prior to extraction in the case when ridge frequency is too low and if CFG_EXTRACT_MULTISCALE is set to 1. This function returns resize factor (if any) calculated based on the ridge distance analysis. This feature is used for better accuracy with children's fingerprints.

C++

```
IDKIT_API int IEngine_GetFingerprintResizeRatio(IENGINE_USER user, int index, int *
resizeRatio);
```

Parameters

`IENGINE_USER user`

[in] An initialized user structure

`int index`

[in] Index of fingerprint that value is get

`int * resizeRatio`

[out] Encodes resize ratio calculated based on average ridge distance applied when creating template. Value 1024 means no resize was done. Otherwise following formula is valid (for both width and height): width_after_resize=resizeRatio*width/1024

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NULLPARAM	resizeRatio is null

2.1.13 IEngine_GetIntermediateImages Function

C++

```
IDKIT_API int IEngine_GetIntermediateImages(const unsigned char * rawImage, int width, int height, unsigned char * skeleton, unsigned char * filtered, unsigned char * binarized, int * bWidth, int * bHeight, unsigned char * bMask);
```

Remarks

Unsupported function for internal purposes

2.1.14 IEngine_GetIrisTemplate Function

Exports iris template at specified index from user structure.

C++

```
IDKIT_API int IEngine_GetIrisTemplate(const IENGINE_USER user, int irisIndex, unsigned char * templateData, int * length);
```

Description

The exported iris template stored in output memory buffer can be re-imported using IEngine_AddIrisTemplate (see page 38).

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`int irisIndex`

[in] Index of particular iris template

`unsigned char * templateData`

[out] Pointer to the memory space where iris template raw data will be saved

`int * length`

[in/out] On input, length parameter is interpreted as total size of allocated memory pointed by templateData. On return, this parameter will be equal to the total length of user template.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified iris index is not valid.
IENGINE_E_NULLPARAM	Parameter length must not be NULL.
IENGINE_E_NOIRIS	User structure contains no iris templates.
IENGINE_E_NOTCONNECTED	IDKit not connected.
IENGINE_E_INTERNAL	Unspecified internal error occurred.

Remarks

This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_AddIrisTemplate (see page 38), IEngine_SetIrisTemplate (see page 39), IEngine_ExportUserTemplate (see page 56), IEngine_ImportUserTemplate (see page 57)

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.1.15 IEngine_GetMinutiaePoints Function

Returns list of fingerpint minutiae points (endings and bifurcations)

C++

```
IDKIT_API int IEngine_GetMinutiaePoints(const IENGINE_USER user, int fingerprintIndex, int
* minutiaeCount, IENGINE_MINUTIAE * minutiae);
```

Parameters

`const IENGINE_USER user`

[in] An initialized user structure

`int fingerprintIndex`

[in] Index of the fingerprint which minutiae list will be returned

`int * minutiaeCount`

[out] On return, this parameter will contain the total number of extracted minutiae

`IENGINE_MINUTIAE * minutiae`

[out] On return, this array will contain complete list of extracted minutiae. This parameter must be initialized on input, it should contain at least MAX_MINUTIAE_COUNT elements.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_BADUSER	User parameter is not valid.
IENGINE_E_BADINDEX	Specified fingerprint index is not valid.
IENGINE_E_NULLPARAM	Parameter minutiae must not be NULL.
IENGINE_E_INTERNAL	Unspecified internal error occurred.
IENGINE_E_NOTCONNECTED	IDKit not connected.

Remarks

This function returns list of minutiae points (endings and bifurcations) for a specified fingerprint. Each minutiae point is represented by its location, angle and type. The index of the fingerprint image which minutiae are returned is specified by the fingerprintIndex parameter. This function uses two-phase buffer retrieval (see page 118) algorithm.

Related Topics

IEngine_GetMinutiaeImage (see page 52), IEngine_SaveMinutiaeImage (see page 53)

2.1.16 IEngine_MatchUsersModalitiesWithIntermediates Function

C++

```
IDKIT_API int IEngine_MatchUsersModalitiesWithIntermediates(const IENGINE_USER user, const
```

```
IENGINE_USER user2, int modalities, int * score, IntermediateMatch * intermediateMatches,
int * intermediateMatchesLength, ModalityScores * modalityScores);
```

Remarks

Unsupported function for internal purposes

2.1.17 IEngine_MatchUsersWithIntermediates Function

C++

```
IDKIT_API int IEngine_MatchUsersWithIntermediates(const IENGINE_USER probeUser, const
IENGINE_USER galleryUser, int * score, IntermediateMatch * intermediateMatches, int *
intermediateMatchesLength, ModalityScores * modalityScores);
```

Remarks

Unsupported function for internal purposes

2.1.18 IEngine_SetTracingSpanContextJson Function

C++

```
IDKIT_API int IEngine_SetTracingSpanContextJson(const char * ctx, unsigned long ctx_length);
```

Remarks

Unsupported function for internal purposes

2.2 Types, Structures, Enumerations

Enumerations

Name	Description
IENGINE_CONFIG (see page 103)	Enumeration defining codes of configuration parameters
IENGINE_FINGER_POSITION (see page 107)	Enumeration defining codes for different finger positions
IENGINE_IMAGE_FORMAT (see page 108)	Enumeration defining codes for different image formats
IENGINE_TEMPLATE_FORMAT (see page 109)	Enumeration defining codes for different fingerprint template (see page 126) formats
IENGINE_FINGERPRINT_CLASS (see page 110)	Fingerprint class (IAFIS specification https://identifyus.org/help/FPinformation.pdf).
IENGINE_EXTRACTOR_ALGORITHM (see page 111)	Fingerprint template extractor algorithm speed and accuracy levels.
IENGINE_MATCHING_MODALITY (see page 112)	Modalities that can be used for matching in function IEngine_MatchUsersModalities (see page 88).
IENGINE_HARDWARE_ID_METHOD (see page 112)	Enumeration defining codes for different hardware ID methods.

Macros

Name	Description
CRYPT_KEY_LENGTH (see page 110)	Defines the key size of AES cipher. In this version, key size of 256 bits is used (256 bits = 32 bytes).
MAX_CRITICAL_POINTS_COUNT (see page 110)	Defines the maximum number of critical minutiae points count within one fingerprint template.

Structures

Name	Description
IENGINE_MINUTIAE (see page 108)	Structure representing a particular minutiae (distinctive fingerprint feature point found in fingerprint skeleton, such as a bifurcation or an ending).

IENGINE_CRITICAL_POINT (see page 109)	Structure representing a particular critical point (core or delta)

Types

Name	Description
IENGINE_COLLECTION (see page 103)	Pointer to a set of user IDs.
IENGINE_CONNECTION (see page 107)	Handle to a database connection.
IENGINE_USER (see page 110)	Pointer to structure with user data.
IENGINE_IRIS_IMAGE_TYPE (see page 111)	Enumeration defining codes for different iris image types
IENGINE_IRIS_POSITION (see page 111)	Enumeration defining codes for different iris positions

2.2.1 IENGINE_COLLECTION Type

Pointer to a set of user IDs.

C++

```
typedef void * IENGINE_COLLECTION;
```

Remarks

This is a container for user IDs. Several functions in IDKit SDK return user IDs in this form. IENGINE_COLLECTION is an opaque type. Application doesn't need to access internal fields. IENGINE_COLLECTION can be manipulated with functions in collection group (see page 63).

2.2.2 IENGINE_CONFIG Enumeration

Enumeration defining codes of configuration parameters

C++

```
typedef enum {
  CFG_BEST_CANDIDATES_COUNT = 0,
  CFG_SIMILARITY_THRESHOLD = 1,
  CFG_RESOLUTION_DPI = 3,
  CFG_MAX_ROTATION = 4,
  CFG_STORE_IMAGES = 5,
  CFG_IDENTIFICATION_SPEED = 6,
  CFG_LOG_LEVEL = 8,
  CFG_MAX_TEMPLATE_SIZE = 10,
  CFG_JPEG2K_COMPRESSION_RATIO = 11,
  CFG_WSQ_BITRATE = 12,
  CFG_DB_IMAGE_FORMAT = 13,
  CFG_LOAD_IMAGES = 14,
  CFG_EXTRACT_CRITICAL_POINTS = 17,
  CFG_EXTRACTOR_ALGORITHM = 18,
  CFG_IFACE_DETECT_FORCED = 19,
  CFG_IFACE_IGNORE_MULTIPLE_FACES = 23,
  CFG_IFACE_DETECTION_MODE = 24,
  CFG_IFACE_EXTRACTION_MODE = 26,
  CFG_TRACING_ENABLED = 27,
  CFG_IFACE_DETECTION_THRESHOLD = 28,
  CFG_IFACE_IRIS_DETECTION_THRESHOLD = 29,
  CFG_IFACE_MODELS_PATH = 30,
  CFG_EXTRACT_MULTISCALE = 31,
  CFG_PALM_QUALITY_UPPER_BOUND = 32,
  CFG_PALM_QUALITY_LOWER_BOUND = 33,
  CFG_IFACE_ONNXRT_INTER_THREADS = 34,
  CFG_IFACE_ONNXRT_INTRA_THREADS = 35
} IENGINE_CONFIG;
```

Description

Parameter settings have impact on the behavior of the library. Use IEngine_GetParameter (see page 93) and

IEngine_SetParameter (see page 93) functions to read/write parameter values. Global parameters, e.g. CFG_LOG_LEVEL, apply to all connections, non-global parameters are unique for a particular connection.

Members

Members	Description
CFG_BEST_CANDIDATES_COUNT = 0	Specifies the number of best candidates returned by Identification functions (see page 72). This parameter has impact on identification algorithm performance. Best strategy is to set best candidates count to 1 and in case of positive match (1 candidate was found) you can repeat the identification with higher candidate count. The assumption is that you get no candidate most of the time. Default value is 1.
CFG_SIMILARITY_THRESHOLD = 1	Specifies the similarity threshold for Identification functions (see page 72) and Verification functions (see page 79). Valid range is between 0 and 1000, where [0, 1000] is closed interval. Default value is 40.
CFG_RESOLUTION_DPI = 3	Specifies the resolution of input fingerprint images in DPI. Default value is 500 DPI.
CFG_MAX_ROTATION = 4	When taking fingerprint images, it is very likely that fingers might not be aligned exactly at the same angle every time. It defines maximum allowed rotation between two fingerprint images, this parameter controls Verification functions (see page 79) and Identification functions (see page 72). Valid range is between 0 and 180, where [0, 180] is closed interval. Default value is 180.
CFG_STORE_IMAGES = 5	Specifies whether input fingerprint images should be stored in the database or not. Value 0 means only fingerprint templates are stored. Value 1 means both templates and images are stored during user registration. Note that this parameter affects only database storage. In-memory copies of IENGINE_USER (see page 110) structure retain image data until they are destroyed. Default value is 1. **Note:** In IDKit Android SDK, IDKit Multi Android SDK and IDKit Embedded SDK, images are stripped immediately after template extraction (IEngine_AddFingerprint (see page 14) and similar functions) and in-memory IENGINE_USER (see page 110) structures do not contain images if this parameter is set to 0. Default value for IDKit Android SDK, IDKit Multi Android SDK and IDKit Embedded SDK is 0.
CFG_IDENTIFICATION_SPEED = 6	Specifies the speed at which Identification functions (see page 72) operate. Valid parameters range is 0-4. Value 1 means the slowest and the most precise algorithm. Value 4 selects the fastest algorithm. Value 0 automatically selects reasonable identification speed for the number of fingerprints searched. For more information about different speed levels, please refer to Identification speed (see page 4). Default value is 0 what means speed 4 is used. **Note:** Identification speed can be set only for fingerprint identification. For face and iris identification always the same algorithm is used.
CFG_LOG_LEVEL = 8	Controls amount of information that appears in logs. See function IEngine_SetLogFile (see page 95) for ways to redirect log messages to a file. Value -1 means no log messages, value 0 only error messages, value 1 significant events and reports of current state, and value 2 some extra debugging information. Value 3 enables debugging output of licence validation process. Global parameter - applies to all connections. Default is 0.
CFG_MAX_TEMPLATE_SIZE = 10	Defines maximum fingerprint template (see page 126) size in bytes. When this parameter is zero, template size is unrestricted. When this parameter is non-zero, template extraction (see page 136) will trim larger templates down to the size specified by this parameter. The parameter is used during fingerprint extraction. The limit doesn't apply on existing templates (User instances), but only on newly extracted templates. The limit is applied per finger view. If User template contains single fingerprint, its size respects the parameter. In multi-finger templates, the limit is increased proportionally to the number of fingerprints in the template. If User template contains N fingerprints, its size is lower or equal than N x CFG_MAX_TEMPLATE_SIZE. This parameter applies only to Innovatrics proprietary templates. Standard ANSI/ISO templates are not affected. Valid value range is 1024 to 65500. Default value is 0, unlimited template size.
CFG_JPEG2K_COMPRESSION_RATIO = 11	Defines compression ratio for JPEG2K compression algorithm. Used compression algorith supports minimal compression ratio 1:10. Valid value range is 1 to infinite. Default value is 15, what corresponds to 1:15 compression ratio.

CFG_WSQ_BITRATE = 12	Defines bitrate for WSQ compression algorithm. Native bitrate for algorithm is calculated as CFG_WSQ_BITRATE/100. Please notice that there is no universal correlation between bitrate and compression ratio because it depends on image content also. Bitrate is defined as average bits per pixel in the compressed image. Valid value range is 1 to 500. Default value is 125 (native is 125/100 = 1.25), what corresponds approximately to 1:10 compression ratio for standard fingerprint image.
CFG_DB_IMAGE_FORMAT = 13	Defines format of images stored in the database. Default image format is PNG (BMP for IDKit Android SDK and IDKit Embedded SDK). You can use all supported image formats. Valid values are listed in enumeration IENGINE_IMAGE_FORMAT (see page 108).
CFG_LOAD_IMAGES = 14	Specifies whether fingerprint images should be loaded from the database or not (during IEngine_GetUser (see page 69)). Value 0 means only fingerprint templates are loaded. Value 1 means both templates and images are loaded into user record during user load from database.
CFG_EXTRACT_CRITICAL_POINTS = 17	Enables/Disables extraction of critical points (IENGINE_CRITICAL_POINT (see page 109), IEngine_GetDeltasAndCores (see page 51)) from fingerprint image during template extraction (see page 136). Please notice that critical points extraction is time consuming operation and it has impact on extraction duration. Default value is 1 (enabled). **Note for IDKit Android SDK and IDKit Embedded SDK:** Default value is 0 (disabled).
CFG_EXTRACTOR_ALGORITHM = 18	Allows you to specify speed of extraction algorithm. Valid options are listed in enumeration IENGINE_EXTRACTOR_ALGORITHM (see page 111). Please note that higher speed means lower accuracy, the best extractor accuracy is achieved with speed NORMAL. **Note:** For IDKit PC SDK default is NORMAL, for IDKit Android SDK and IDKit Embedded SDK default value is FASTEST. For embedded and mobile platforms, one possibility how to maximize the accuracy while maintaining good speed is to use NORMAL extractor speed at enrollment and FAST or FASTEST speed at all subsequent verifications (unlike enrollment operation, verification must guarantee response within strict time constraints).
CFG_IFACE_DETECT_FORCED = 19	Enables/Disables IFace detect forced. (This feature is not available on Android.) Value 0 specifies that the detect forced is disabled. Value 1 enables the detect forced. Default value is 0, the detect forced is enabled.
CFG_IFACE_IGNORE_MULTIPLE_FACES = 23	This parameter adjusts the behavior of IDKit in case of occurence multiple faces in the image. Value 0 means that error IENGINE_E_TOOMANYFACES is returned if input image contains more than one face. If value 1 is set, face with maximum score is captured and IENGINE_E_NOERROR is returned even if multiple faces are present in input image. Default value is 0, multi-faces images are not allowed.
CFG_IFACE_DETECTION_MODE = 24	This parameter adjusts face/iris detection speed. Value 0 means that IFace parameter IFACE_PARAMETER_FACEDET_SPEED_ACCURACY_MODE is set to IFACE_FACEDET_SPEED_ACCURACY_MODE_FAST Value 0 means that IFace parameter IFACE_PARAMETER_IRIS_DETECTION_SPEED_ACCURACY_MODE is set to IFACE_IRIS_DETECTION_SPEED_ACCURACY_MODE_FAST Value 1 means that IFace parameter IFACE_PARAMETER_FACEDET_SPEED_ACCURACY_MODE is set to IFACE_FACEDET_SPEED_ACCURACY_MODE_BALANCED Value 1 means that IFace parameter IFACE_PARAMETER_IRIS_DETECTION_SPEED_ACCURACY_MODE is set to IFACE_IRIS_DETECTION_SPEED_ACCURACY_MODE_ACCURATE Value 2 means that IFace parameter IFACE_PARAMETER_FACEDET_SPEED_ACCURACY_MODE is set to IFACE_FACEDET_SPEED_ACCURACY_MODE_ACCURATE Value 2 means that IFace parameter IFACE_PARAMETER_IRIS_DETECTION_SPEED_ACCURACY_MODE is set to IFACE_IRIS_DETECTION_SPEED_ACCURACY_MODE_ACCURATE Value 3 means that IFace parameter IFACE_PARAMETER_FACEDET_SPEED_ACCURACY_MODE is set to IFACE_FACEDET_SPEED_ACCURACY_MODE_ACCURATE_SERVER - Unsupported, for internal purposes. Value 3 means that IFace parameter IFACE_PARAMETER_IRIS_DETECTION_SPEED_ACCURACY_MODE is set to IFACE_IRIS_DETECTION_SPEED_ACCURACY_MODE_ACCURATE Default value on desktop/server platforms is 1. Default value on mobile/embedded platforms is 0.

CFG_IFACE_EXTRACTION_MODE = 26	This parameter adjusts face/iris extraction speed. Value 0 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Value 0 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Value 1 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_FAST Value 1 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE_P1 Value 2 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_ACCURATE_SERVER - Unsupported, for internal purposes. Value 2 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Value 3 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_ACCURATE_SERVER_WILD - Unsupported, for internal purposes. Value 3 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Value 4 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_ACCURATE_SERVER_VISA - Unsupported, for internal purposes. Value 4 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Value 5 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_BALANCED Value 5 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Value 6 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_ACCURATE_SERVER_MASK - Unsupported, for internal purposes. Value 6 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Value 7 means that IFace parameter IFACE_PARAMETER_FACETMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_FACETMPLEXT_SPEED_ACCURACY_MODE_ACCURATE_MASK - Unsupported, for internal purposes. Value 7 means that IFace parameter IFACE_PARAMETER_IRISTMPLEXT_SPEED_ACCURACY_MODE is set to IFACE_IRISTMPLEXT_SPEED_ACCURACY_MODE_ACCURATE Default value on desktop/server platforms is 0. Default value on mobile/embedded platforms is 1.
CFG_TRACING_ENABLED = 27	Unsupported parameter for internal purposes
CFG_IFACE_DETECTION_THRESHOLD = 28	valid values are -1 - 10000 -1 means default detection threshold for given detection mode is used 0-10000 changes detection threshold for currently set detection mode Detection threshold is set for active detection mode.
CFG_IFACE_IRIS_DETECTION_THRESHOLD = 29	Specifies the threshold for iris detection confidence. The threshold MUST be set according to quality of iris dataset and use case. When you want to add irises with lower quality then set threshold towards 0. When you want to add only high quality irises then set threshold towards 10000. This configuration parameter influences fail to enroll and find/match accuracy. In case threshold is set high and iris dataset has lower quality, then fail to enroll is higher. When threshold is lowered on lower quality dataset then fail to enroll is also lower, but find/match accuracy can be lower too. Valid range is [0, 10000]. Default value is 9900.
CFG_IFACE_MODELS_PATH = 30	Specifies path to IFace models. The value is a string and it has to be set before initialization. Use IEngine_SetStringParameter (see page 94) and IEngine_GetStringParameter (see page 94) to access the parameter.

CFG_EXTRACT_MULTISCALE = 31	Enables/Disables ridge distance normalization. This feature is used for better accuracy with children's fingerprints. Default Value 0 means no normalization If value is 1 then rescale fingerprint image prior to extraction in the case when ridge frequency is too low. It is used only when CFG_EXTRACTOR_ALGORITHM is EXTRACTOR_SPEED_FAST or EXTRACTOR_SPEED_FASTEST.
CFG_IFACE_ONNXRT_INTER_THREADS = 34	It has to be set before initialization (before IEngine_InitModule (see page 8)/IEngine_InitWithLicense (see page 8) call). Parameter defining the number of threads used to parallelize the execution of the graph (across nodes). Use IEngine_SetStringParameter (see page 94) and IEngine_GetStringParameter (see page 94) to access the parameter or set environment variable IFACE_ONNXRT_INTER_THREADS. Once environment variable is set, it overrides values set by function IEngine_SetStringParameter (see page 94). The value MUST be positive integer, if 0 is set, default onnxruntime thread count is used. Default value is 0. This parameter sets the number of inter threads for all solvers.
CFG_IFACE_ONNXRT_INTRA_THREADS = 35	It has to be set before initialization (before IEngine_InitModule (see page 8)/IEngine_InitWithLicense (see page 8) call). Parameter defining the number of threads to use to run the solver. Use IEngine_SetStringParameter (see page 94) and IEngine_GetStringParameter (see page 94) to access the parameter or set environment variable IFACE_ONNXRT_INTRA_THREADS. Once environment variable is set, it overrides values set by function IEngine_SetStringParameter (see page 94). The value MUST be positive integer, if 0 is set, default onnxruntime thread count is used. Default value is 0. This parameter sets the number of intra threads for all solvers.

Compatibility

The following parameters are supported in all versions of IDKit SDK. Parameter CFG_STORE_IMAGES has different behavior and default value in IDKit Android SDK, IDKit Multi Android SDK and IDKit Embedded SDK. See comments for CFG_STORE_IMAGES above.

2.2.3 IENGINE_CONNECTION Type

Handle to a database connection.

C++

```
typedef void * IENGINE_CONNECTION;
```

Remarks

This handle represents a single database connection (see page 119). It is created with IEngine_InitConnection (see page 9) and deleted with IEngine_CloseConnection (see page 11). It can be selected for current thread with IEngine_SelectConnection (see page 10).

Availability

Supported in IDKit PC SDK and IDKit Multi SDK only. IDKit Android SDK, IDKit Multi Android SDK and IDKit Embedded SDK don't support multiple simultaneous connections.

2.2.4 IENGINE_FINGER_POSITION Enumeration

Enumeration defining codes for different finger positions

C++

```
typedef enum {
  UNKNOWN_FINGER = 0,
  RIGHT_THUMB = 1,
  RIGHT_INDEX = 2,
  RIGHT_MIDDLE = 3,
  RIGHT_RING = 4,
```

```
  RIGHT_LITTLE = 5,
  LEFT_THUMB = 6,
  LEFT_INDEX = 7,
  LEFT_MIDDLE = 8,
  LEFT_RING = 9,
  LEFT_LITTLE = 10,
  UNKNOWN_PALM = 20,
  RIGHT_FULL_PALM = 21,
  LEFT_FULL_PALM = 23,
  UNKNOWN_PRINT = 40
} IENGINE_FINGER_POSITION;
```

2.2.5 IENGINE_IMAGE_FORMAT Enumeration

Enumeration defining codes for different image formats

C++

```
typedef enum {
  IENGINE_FORMAT_BMP = 0,
  IENGINE_FORMAT_PNG = 1,
  IENGINE_FORMAT_JPG = 2,
  IENGINE_FORMAT_GIF = 3,
  IENGINE_FORMAT_TIF = 4,
  IENGINE_FORMAT_WSQ = 5,
  IENGINE_FORMAT_JPEG2K = 6,
  IENGINE_FORMAT_LAST_DUMMY = 7
} IENGINE_IMAGE_FORMAT;
```

Availability

Fully supported in IDKit PC SDK and IDKit Multi SDK. IDKit Android SDK and IDKit Multi Android SDK support RAW, WSQ, BMP and JPEG2K format.

2.2.6 IENGINE_MINUTIAE Structure

Structure representing a particular minutiae (distinctive fingerprint feature point found in fingerprint skeleton, such as a bifurcation or an ending).

C++

```
typedef struct {
  unsigned char angle;
  unsigned short x;
  unsigned short y;
  unsigned char type;
  unsigned char quality;
} IENGINE_MINUTIAE;
```

Members

Members	Description
`unsigned char angle;`	Minutia angle encoded in one byte. Valid range: 0-255. One unit represents 360/256 = 1.40625 degrees.
`unsigned short x;`	Minutiae (see page 49) x coordinate as stored in the template.
`unsigned short y;`	Minutiae (see page 49) y coordinate as stored in the template.
`unsigned char type;`	Minutiae (see page 49) type: 0=bifurcation, 1=ending
`unsigned char quality;`	Minutiae (see page 49) quality (0.. undefined, 1 - lowest, 255 - highest)

2.2.7 IENGINE_CRITICAL_POINT Structure

Structure representing a particular critical point (core or delta)

C++

```
typedef struct {
  unsigned char angle;
  unsigned short x;
  unsigned short y;
  unsigned char type;
} IENGINE_CRITICAL_POINT;
```

Members

Members	Description
`unsigned char angle;`	Minutia angle encoded in one byte. Valid range: 0-255.
`unsigned short x;`	Minutiae (see page 49) x coordinate as stored in the template.
`unsigned short y;`	Minutiae (see page 49) y coordinate as stored in the template.
`unsigned char type;`	Critical point type (core = 1, delta = 2)

2.2.8 IENGINE_TEMPLATE_FORMAT Enumeration

Enumeration defining codes for different fingerprint template (see page 126) formats

C++

```
typedef enum {
  FORMAT_ICS = 1,
  FORMAT_ANSI = 2,
  FORMAT_ISO = 3,
  FORMAT_ANSI_PLUS = 4,
  FORMAT_ISO_PLUS = 5
} IENGINE_TEMPLATE_FORMAT;
```

Members

Members	Description
`FORMAT_ICS = 1`	Innovatrics proprietary template format containing both minutiae and pattern information. It provides maximum recognition accuracy and fastest 1:N identification. Exact template version is stored in initial 6 characters of template data. Only ICRS23 templates are supported. Backward compatibility with templates of older format (ICRS20, ICRS21 or ICRS22) is not supported. To import or load such templates, convert them into ICRS23 format using IDKit SDK
`FORMAT_ANSI = 2`	Standard ANSI/INCITS 378 minutiae template. It can be used in Innovatrics ANSI/ISO SDK and in other ANSI/INCITS 378 compliant software. This template format can be exported from IDKit SDK, but it cannot be imported back. Consider FORMAT_ANSI_PLUS if you need import functionality. During export of user template in this format face and iris templates are trimmed so they are not included in exported template.
`FORMAT_ISO = 3`	Standard ISO/IEC 19794-2 minutiae template. It can be used in Innovatrics ANSI/ISO SDK and in other ISO/IEC 19794-2 compliant software. This template format can be exported from IDKit SDK, but it cannot be imported back. Consider FORMAT_ISO_PLUS if you need import functionality. During export of user template in this format face and iris templates are trimmed so they are not included in exported template.
`FORMAT_ANSI_PLUS = 4`	Standard ANSI/INCITS 378 minutiae template with attached Innovatrics proprietary data. It can be used in Innovatrics ANSI/ISO SDK and in other ANSI/INCITS 378 compliant software. Templates in this format are larger than templates in FORMAT_ANSI format, but they can be imported back to IDKit SDK just like Innovatrics proprietary templates. During export of user template in this format face and iris templates are trimmed so they are not included in exported template.

FORMAT_ISO_PLUS = 5	Standard ISO/IEC 19794-2 minutiae template with attached Innovatrics proprietary data. It can be used in Innovatrics ANSI/ISO SDK and in other ISO/IEC 19794-2 compliant software. Templates in this format are larger than templates in FORMAT_ISO format, but they can be imported back to IDKit SDK just like Innovatrics proprietary templates. During export of user template in this format face and iris templates are trimmed so they are not included in exported template.

2.2.9 IENGINE_USER Type

Pointer to structure with user data.

C++

```
typedef void * IENGINE_USER;
```

Description

This pointer points to a structure with user data. The pointed structure is used internally and should not be accessed or manipulated directly, only through User related functions (see page 11).

Related Topics

IEngine_InitUser (see page 12), User related functions (see page 11)

2.2.10 CRYPT_KEY_LENGTH Macro

C++

```
#define CRYPT_KEY_LENGTH 32
```

Remarks

Defines the key size of AES cipher. In this version, key size of 256 bits is used (256 bits = 32 bytes).

2.2.11 MAX_CRITICAL_POINTS_COUNT Macro

C++

```
#define MAX_CRITICAL_POINTS_COUNT 16
```

Remarks

Defines the maximum number of critical minutiae points count within one fingerprint template.

2.2.12 IENGINE_FINGERPRINT_CLASS Enumeration

Fingerprint class (IAFIS specification https://identifyus.org/help/FPinformation.pdf).

C++

```
typedef enum {
  UNKNOWN = 0,
  LEFT_LOOP = 1,
  RIGHT_LOOP = 2,
  ARCH = 3,
  WHORL = 4
} IENGINE_FINGERPRINT_CLASS;
```

Description

This enumeration is used with IEngine_GetFingerprintClass (see page 99) function.

Members

Members	Description
UNKNOWN = 0	Unknown class.
LEFT_LOOP = 1	Left loop class.
RIGHT_LOOP = 2	Right loop class.
ARCH = 3	Arch class.
WHORL = 4	Whorl class.

2.2.13 IENGINE_EXTRACTOR_ALGORITHM Enumeration

Fingerprint template extractor algorithm speed and accuracy levels.

C++

```
typedef enum {
  EXTRACTOR_SPEED_NORMAL = 0,
  EXTRACTOR_SPEED_FAST = 2,
  EXTRACTOR_SPEED_FASTEST = 4,
  EXTRACTOR_SPEED_AFIS = 8
} IENGINE_EXTRACTOR_ALGORITHM;
```

Description

This enumeration is used with parameter CFG_EXTRACTOR_ALGORITHM and functions IEngine_GetParameter (see page 93) and IEngine_SetParameter (see page 93).

Members

Members	Description
EXTRACTOR_SPEED_NORMAL = 0	Extracting algorithm will be optimized for accuracy. This is default extraction mode for IDKit SDK on PC platform.
EXTRACTOR_SPEED_FAST = 2	Compromise between speed and accuracy of extraction algorithm.
EXTRACTOR_SPEED_FASTEST = 4	Fastest mode of extraction algorithm but accuracy may be affected. This is default setting for IDKit Android SDK, IDKit Multi Android SDK and IDKit Embedded SDK versions.
EXTRACTOR_SPEED_AFIS = 8	Unsupported, for internal purposes.

2.2.14 IENGINE_IRIS_IMAGE_TYPE Type

Enumeration defining codes for different iris image types

C++

```
typedef enum IENGINE_IRIS_IMAGE_TYPE@1 IENGINE_IRIS_IMAGE_TYPE;
```

Availability

Supported in IDKit Multi SDK.

2.2.15 IENGINE_IRIS_POSITION Type

Enumeration defining codes for different iris positions

C++

```
typedef enum IENGINE_IRIS_POSITION@1 IENGINE_IRIS_POSITION;
```

Availability

Supported in IDKit Multi SDK.

2.2.16 IENGINE_MATCHING_MODALITY Enumeration

Modalities that can be used for matching in function IEngine_MatchUsersModalities (see page 88).

C++

```
typedef enum {
  MODALITY_FINGER = 1<<0,
  MODALITY_FACE = 1<<1,
  MODALITY_IRIS = 1<<3,
  MODALITY_ALL = 0xFFFFFFFF
} IENGINE_MATCHING_MODALITY;
```

Description

This enumeration is used for selection of modalities used for matching users in function IEngine_MatchUsersModalities (see page 88). Modalities can be combined with operator '|' (bitwise or), e.g. 'MODALITY_FINGER|MODALITY_FACE' selects finger and face for matching. MODALITY_ALL selects all supported modalities and it is not needed to be combined with another modalities.

Members

Members	Description
MODALITY_FINGER = 1<<0	Use finger print modality for matching. MODALITY_FINGER corresponds to value 1.
MODALITY_FACE = 1<<1	Use facial modality for matching. MODALITY_FACE corresponds to value 2.
MODALITY_IRIS = 1<<3	Use iris modality for matching. MODALITY_IRIS corresponds to value 8.
MODALITY_ALL = 0xFFFFFFFF	Use all supported modalities for matching. MODALITY_ALL corresponds to value 256*256*256*256-1.

Availability

Supported in IDKit Multi SDK and IDKit Multi Android SDK.

2.2.17 IENGINE_HARDWARE_ID_METHOD Enumeration

Enumeration defining codes for different hardware ID methods.

C++

```
typedef enum {
  IENGINE_HARDWARE_ID_METHOD_AUTO = 0,
  IENGINE_HARDWARE_ID_METHOD_DISKID = 1,
  IENGINE_HARDWARE_ID_METHOD_MAC = 2,
  IENGINE_HARDWARE_ID_METHOD_SERIALNO = 3,
  IENGINE_HARDWARE_ID_METHOD_IMEI = 4,
  IENGINE_HARDWARE_ID_METHOD_SMBIOS = 5,
  IENGINE_HARDWARE_ID_METHOD_AMAZON = 6,
  IENGINE_HARDWARE_ID_METHOD_APPID = 7,
  IENGINE_HARDWARE_ID_METHOD_PHY = 8,
  IENGINE_HARDWARE_ID_METHOD_MAC_ADDR = 9,
  IENGINE_HARDWARE_ID_METHOD_SYS = 10,
  IENGINE_HARDWARE_ID_METHOD_ANDROID_ID = 11
} IENGINE_HARDWARE_ID_METHOD;
```

Description

This enumeration is used in IEngine_GetHardwareIdByMethod (see page 98)function

Members

Members	Description
IENGINE_HARDWARE_ID_METHOD_AUTO = 0	Automatically selects the hardware ID method based on platform defaults On Linux platforms, IENGINE_HARDWARE_ID_METHOD_MAC is being used On Windows platforms, IENGINE_HARDWARE_ID_METHOD_DISKID is being used On Android platforms, IENGINE_HARDWARE_ID_METHOD_APPID is being used
IENGINE_HARDWARE_ID_METHOD_DISKID = 1	Use disk based hardware id This method is available only on Windows platforms
IENGINE_HARDWARE_ID_METHOD_MAC = 2	Use network MAC address based hardware ID This method is available only on Linux platforms
IENGINE_HARDWARE_ID_METHOD_SERIALNO = 3	Use serial number based hardware ID This method is available only on older Android OS versions (Android OS version < 7.2.1), and requires appropriate permissions
IENGINE_HARDWARE_ID_METHOD_IMEI = 4	Use IMEI based hardware ID This method is available only on older Android OS versions (Android OS version < 7.2.1), and required appropriate permissions
IENGINE_HARDWARE_ID_METHOD_SMBIOS = 5	Use SMBIOS UUID based hardware ID Not supported
IENGINE_HARDWARE_ID_METHOD_AMAZON = 6	Use Amazon instance ID based hardware ID Not supported
IENGINE_HARDWARE_ID_METHOD_APPID = 7	Use application ID based hardware ID Available on Android platforms
IENGINE_HARDWARE_ID_METHOD_PHY = 8	Use physical address based hardware ID Not Supported
IENGINE_HARDWARE_ID_METHOD_MAC_ADDR = 9	Use only network mac address as hardware id Available on ARM/Android ("IM00" form)
IENGINE_HARDWARE_ID_METHOD_SYS = 10	Use content of a system file to generate hardware id (/proc, /sys on Linux systems)
IENGINE_HARDWARE_ID_METHOD_ANDROID_ID = 11	Use android ID. The ANDROID_ID property is a 64-bit number that is generated when the device is first booted, and is supposed to remain constant for the lifetime of the device.

2.3 Function Quick Reference

Function	Summary
IEngine_AddFingerprint (see page 14)	Adds fingerprint to user structure, fingerprint image is read from memory
IEngine_AddFingerprintFromFile (see page 17)	Adds fingerprint to user structure, fingerprint image is read from file
IEngine_AttachFingerprintImage (see page 44)	Links fingerprint image with a particular fingerprint
IEngine_ClearDatabase (see page 70)	Deletes all entries from database
IEngine_ClearTag (see page 61)	Removes tag from the user.
IEngine_ClearUser (see page 13)	Clears all information from user structure
IEngine_CloseConnection (see page 11)	Closes a connection and releases all resources held by the connection.
IEngine_Connect (see page 10)	Connects to a database
IEngine_ConvertRawImage2Bmp	Converts raw image to BMP image
IEngine_ConvertBmp2RawImage	Converts BMP image to raw image
IEngine_DeserializeUser (see page 56)	Converts a sequence of bytes into the corresponding user structure
IEngine_ExportUserTemplate (see page 56)	Exports user template
IEngine_FindFingerprint (see page 75)	Identifies particular fingerprint in the database
IEngine_FindFingerprintByQuery (see page 77)	Identifies particular fingerprint in a subset of the database defined by query.
IEngine_FindFingerprintInMemory (see page 78)	Identifies particular fingerprint in an external database

IEngine_FindFingerprintInSelection (see page 76)	Identifies particular fingerprint in a sub-part of the database
IEngine_FindUser (see page 72)	Identifies user in the database
IEngine_FindUserByQuery (see page 74)	Identifies user in a subset of the database defined by query.
IEngine_FindUserInMemory (see page 74)	Identifies user in an external database
IEngine_FindUserInSelection (see page 73)	Identifies user in a sub-part of the database
IEngine_FreeCollection (see page 64)	Releases memory held by IENGINE_COLLECTION (see page 103).
IEngine_FreeUser (see page 13)	Releases user structure
IEngine_GetAllUserIDs (see page 71)	Retrieves list of all user IDs in the database.
IEngine_GetCollectionIDs (see page 64)	Retrieves IDs from a collection.
IEngine_GetCollectionSize (see page 64)	Gets the number of IDs in the collection.
IEngine_GetCustomData (see page 49)	Retrieves application specific data attached to the user
IEngine_GetErrorMsg (see page 97)	Returns error message formated as string
IEngine_GetFingerPosition (see page 41)	Returns finger position of fingerprint at specified index
IEngine_GetFingerprintCount (see page 41)	Returns total number of fingerprints contained in user structure
IEngine_GetFingerprintImage (see page 43)	Returns fingerprint image from user structure, image data is written to memory
IEngine_GetFingerprintQuality (see page 51)	Returns quality of fingerprint at specified index
IEngine_GetIntTag (see page 59)	Gets tag value as an integer.
IEngine_GetMinutiaeImage (see page 52)	Returns fingerprint minutiae points overlapped on the original fingerprint image, image is returned in memory
IEngine_GetMinutiaePoints (see page 101)	Returns list of fingerpint minutiae points (endings and bifurcations)
IEngine_GetParameter (see page 93)	Gets parameter value
IEngine_GetStringTag (see page 60)	Gets tag value as a string.
IEngine_GetTagCount (see page 62)	Gets the total number of tags set on a user.
IEngine_GetTagName (see page 62)	Gets name of a tag at the specified index.
IEngine_GetUser (see page 69)	Retrieves user data from database
IEngine_GetUserCount (see page 70)	Returns total number of users registered in database
IEngine_GetUserIDsByQuery (see page 71)	Retrieves list of user IDs in the database that match tag query.
IEngine_GetUserLimit (see page 98)	Returns maximum allowed number of users
IEngine_GetVersion	Deprecated. Returns the library version.
IEngine_GetVersionInfo	Returns the library version
IEngine_HasTag (see page 61)	Checks whether user has a tag set.
IEngine_ImportUserTemplate (see page 57)	Imports fingerprint templates from user template
IEngine_InitClient	Initializes the library with the software license from central Innovatrics Matcher.
IEngine_InitCollection (see page 63)	Initializes collection of user IDs.
IEngine_InitConnection (see page 9)	Initializes new connection to database or ExpressID AFIS.
IEngine_InitModule (see page 8)	Initializes the library
IEngine_InitUser (see page 12)	Initializes an empty user structure
IEngine_MatchFingerprint (see page 79)	Compares a particular fingerprint with fingerprints of another user
IEngine_MatchFingerprints (see page 80)	Compares two fingerprints

IEngine_MatchUser (see page 86)	Compares fingerprints of the input user with fingerprints of a particular user registered in the database.
IEngine_MatchUsers (see page 87)	Compares two users based on their fingerprints
IEngine_RegisterUser (see page 65)	Adds user to database
IEngine_RegisterUserAs (see page 66)	Adds user to database and sets its id number
IEngine_RemoveFingerprint (see page 21)	Removes fingerprint at specified index
IEngine_RemoveUser (see page 69)	Removes user from database
IEngine_SaveFingerprintImage (see page 45)	Returns fingerprint image from user structure, saves the result to a file
IEngine_SaveMinutiaeImage (see page 53)	Saves fingerprint minutiae points overlapped on the original fingerprint image
IEngine_SelectConnection (see page 10)	Selects active connection for the current thread.
IEngine_SerializeUser (see page 55)	Converts user structure into a sequence of bytes
IEngine_SetCryptKey (see page 95)	Sets the cipher key used for the encryption of database content
IEngine_SetCustomData (see page 48)	Attaches the application specific data to the user
IEngine_SetFingerPosition (see page 42)	Sets finger position of fingerprint at specified index
IEngine_SetFingerprint (see page 18)	Replaces fingerprint at specified index, fingerprint image is read from memory
IEngine_SetFingerprintFromFile (see page 20)	Replaces fingerprint at specified index, fingerprint image is read from file
IEngine_SetIntTag (see page 58)	Sets tag value to the specified integer value.
IEngine_SetParameter (see page 93)	Sets parameter value
IEngine_SetStringTag (see page 59)	Sets tag value to the specified string.
IEngine_TerminateModule (see page 9)	Terminates the use of the library
IEngine_UpdateUser (see page 67)	Updates user data contained in database

2.4 Error Codes

Error codes greater and equal to 50000 are Innovatrics license check related errors and you can get detailed license error message using IEngine_GetErrorMsg (see page 97) API call. Error codes greater and equal to 70000 are Innovatrics face modality related errors and you can get detailed face modality error message using IEngine_GetErrorMsg (see page 97) API call. Error codes greater and equal to 60000 are Innovatrics iris modality related errors and you can get detailed iris modality error message using IEngine_GetErrorMsg (see page 97) API call.

Error	Error Code	Description
IENGINE_E_NOERROR	0	No error.
IENGINE_E_BADPARAM	1101	Invalid configuration parameter.
IENGINE_E_NOFINGERPRINT	1102	User structure contains no fingerprints (void user).
IENGINE_E_DBOPEN	1111	Could not connect to database.
IENGINE_E_DBFAILED	1112	Unexpected database failure occurred.
IENGINE_E_DBACCESSDENIED	1113	Database file access is denied.

IENGINE_E_BLANKIMAGE	1114	Image is blank or contains non-recognizable fingerprint.
IENGINE_E_BADIMAGE	1115	Invalid image or unsupported image format.
IENGINE_E_INIT	1116	Library was not initialized.
IENGINE_E_FILE	1117	Error occurred while opening/accessing file.
IENGINE_E_BADUSER	1118	Input user parameter is not valid.
IENGINE_E_BADINDEX	1119	Fingerprint index is not valid.
IENGINE_E_MEMORY	1120	Memory allocation failed.
IENGINE_E_NULLPARAM	1121	Null input parameter provided.
IENGINE_E_OTHER	1122	Other unspecified error.
IENGINE_E_NOIMAGE	1123	Image not available.
IENGINE_E_INTERNAL	1124	Unspecified internal error occurred.
IENGINE_E_NONEXISTINGID	1125	User id not found in database.
IENGINE_E_DUPLICATEID	1126	User id already exists.
IENGINE_E_BADUSERID	1127	User id is not valid.
IENGINE_E_DBFULL	1128	Exceeded database user limit.
IENGINE_E_BADLICENSE	1129	License is not valid, or no license was found. (You can set logging level to 3 in order to see license validation debugging output)
IENGINE_E_EXPIREDLICENSE	1130	License has expired.
IENGINE_E_MISSINGDLL	1131	At least one required DLL could not be loaded.
IENGINE_E_BADFORMAT	1132	Unsupported format.
IENGINE_E_BADVALUE	1133	Invalid value provided.
IENGINE_E_BADTEMPLATE	1135	Invalid template or unsupported template format.
IENGINE_E_QUERYSYNTAX	1136	Syntax error in tag query.
IENGINE_E_INCOMPATIBLE_TEMPLATE	1137	Input template is incompatible with current version and its enabled features.
IENGINE_E_BADCRYPTKEY	1140	Invalid encryption key.
IENGINE_E_SSL	1141	Unable to encrypt communication link with SSL.
IENGINE_E_USERFULL	1142	Maximum fingerprint count in one user record is 255.
IENGINE_E_DBMISSINGTABLE	1143	Some tables are missing in the database.
IENGINE_E_DBBADVERSION	1144	Actual database version differs from the expected version.
IENGINE_E_INTDBFULL	1145	Internal DB is full.
IENGINE_E_SMALLIMAGE	1146	Image is too small. Minimal size of image must be 90x90.
IENGINE_E_BIGIMAGE	1147	Image is too big. Maximal size of image must be 4000x4000.
IENGINE_E_BADDPI	1148	Image does not have required DPI or bitrate.
IENGINE_E_TOOMANYFACES	1149	Image contains too many faces.
IENGINE_E_NOTIMPLEMENTED	1150	Operation is not implemented in this IDKit product.
IENGINE_E_BADMASK	1151	Image mask is not in correct format ([0,255] array).
IENGINE_E_NOTEMPLATES	1160	User structure contains no templates (void user).
IENGINE_E_NOFACES	1161	User structure contains no face templates (void user).
IENGINE_E_NOIRIS	1163	User structure contains no iris templates (void user).
IENGINE_E_SOME_TEMLATE_DATA_MISSING	1164	Some additional data is missing in a modality template.
IENGINE_E_CONSTR	1202	Connection string format not recognized.

IENGINE_E_CONTYPE	1203	Invalid connection type.
IENGINE_E_NOTCONNECTED	1204	IDKit not connected. unavailable.
IENGINE_E_NOTSUPPORTED	1211	This IDKit call is not supported for this connection type.
IENGINE_E_PLUGIN_CANNOT_LOAD	1300	IDKit plugin cannot be loaded.
IENGINE_E_PLUGIN_UNKNOWN	1301	IDKit plugin unknown error (see log file).
IENGINE_E_PLUGIN_NOT_FOUND	1302	IDKit plugin was not found (no such registered plugin).
IENGINE_E_PLUGIN_DUPLICATEID	1303	IDKit plugin already exists (duplicate plugin UID).
IENGINE_E_PLUGIN_UID_INCONSISTENCY	1304	IDKit plugin UID is inconsistent (not constant) on ExpressID AFIS Government Nodes.
IENGINE_E_UNKNOWN_MSG	other	Unknown error

3 Notes

3.1 Buffers (C++)

C++ API functions in IDKit SDK use two-phase buffer retrieval algorithm.

Remarks

Some functions in C++ API return their result in two phases. Application calls these functions twice. First time only length of the buffer is retrieved. Second call then retrieves buffer data. This behavior is intended to give the application a chance to allocate a buffer of sufficient size.

IDKit SDK functions differentiate between the two calls by value of the length parameter. If length is zero or too small or if the data buffer is NULL, only length is returned. If buffer is not NULL and length is sufficient, data is written into the buffer.

Related Topics

IEngine_GetFingerprintImage (see page 43)

IEngine_GetCustomData (see page 49)

IEngine_GetMinutiaePoints

IEngine_GetMinutiaeImage (see page 52)

IEngine_SerializeUser (see page 55)

IEngine_ExportUserTemplate (see page 56)

IEngine_GetStringTag (see page 60)

IEngine_GetTagName (see page 62)

IEngine_GetCollectionIDs (see page 64)

IEngine_ConvertRawToImage (see page 89)

IEngine_ConvertImageToRaw (see page 90)

Example

The following example shows how to retrieve fingerprint image into a buffer:

```
int length = 0;
int err = IEngine_GetFingerprintImage(user, 0, IENGINE_FORMAT_BMP, NULL, &length);
if (err) return -1;
unsigned char *data = new unsigned char[length];
err = IEngine_GetFingerprintImage(user, 0, IENGINE_FORMAT_BMP, data, &length);
if (err) return -1;
```

Notice how the first call passes NULL for data and 0 for length to indicate that only length of the output buffer should be calculated. The second call then retrieves the image into application-allocated buffer.

3.2 Connections

IDKit SDK supports concurrent database connections and several sets of global parameters.

Remarks

There is always one default connection. This connection is used in applications that do not need multiple connections. IDKit SDK however supports construction of several connections (execution contexts or sessions). This feature is useful in several types of applications:

- Multi-threaded applications need thread-local parameters (like DPI or maximum rotation).
- Multi-threaded application servers need one connection per client session, non-blocking concurrency between connections, and thread-local parameters.
- Applications that use multiple databases need to be able to access them all without repeatedly reconnecting to the various databases.

Connections in IDKit SDK are implemented with the following API:

- C++: IEngine_InitConnection (see page 9), IEngine_SelectConnection (see page 10), IEngine_CloseConnection (see page 11)
- .NET: see IDKit.Connection Class in .NET documentation
- Java: IDKit constructor, IDKit.selectConnection and IDKit.close in Java

Connections exist independently from threads (see page 136). Every thread can access every connection. Operations on the same connection are safely serialized by internal locks. There can be only one direct connection per database (SQLite).

Licensed user limit (see page 130) applies to the sum of sizes of all directly connected databases.

Compatibility

Supported in IDKit PC SDK and IDKit Multi SDK.

3.3 Connection String

Specifies location of biometric database.

Remarks

Connection Types

- **SQLite** - Connection to a flat-file database file contained on a local computer.
- **Memory** - Virtual database that exists only in memory.

Note

Since IDKit SDK version 8, ODBC and JDBC databases and ExpressID AFIS service connection are not available and unsupported.

Syntax

The connection type is determined by connection string. Connection string consists of parameter-value pairs followed by semi-colon ';'.

```
[<param>=<value>;]*
```

Values in connection string are allowed to be enclosed in quotation marks to prevent parsing errors if value contains characters ';' and '='. For an example, consider the following connection string:

```
type=sqlite; file=idkit.db;
```

SQLite

This type of connection uses SQLite database file contained on a local computer. If the specified database file does not exist, it is automatically created.

Parameter	Value
type	For SQLite database specify "sqlite".
file	File name of the database file. It may contain the directory path together with the file name. Specified directory path has to exist. File name may contain Unicode characters (encoded in UTF8 in C++).

Example

In order to connect to a SQLite database contained in a file named idkit.db, the connection string has to be on of the following:

```
type=sqlite; file=idkit.db;
```

Simplified form that automatically assumes the file is in SQLite format:

```
idkit.db
```

Memory

Memory database is a virtual database that stores all biometric data including images in memory. There is one memory database per connection (see page 119). All data is lost when IEngine_TerminateModule (see page 9) or IEngine_CloseConnection (see page 11) is called. If many user records are to be stored, it is advised to turn off image storage (using CFG_STORE_IMAGES parameter, ConnectionParameter.CfgStoreImages in .NET or InstanceParameter.CFG_STORE_IMAGES in Java).

Parameter	Value
type	For memory database connection, specify "memory".

Example

```
type=memory
```

Compatibility

Since IDKit SDK version 8, ODBC and JDBC databases and ExpressID AFIS service connection are not available and unsupported.

3.4 Corresponding Fingerprints

Two fingerprints are considered to correspond to each other if they have the same finger position or if at least one of them has unknown finger position.

Remarks

Only corresponding fingerprints are compared during identification (see page 129) and verification (see page 137). You can use functions IEngine_SetFingerPosition (see page 42) and IEngine_GetFingerPosition (see page 41) in C++, IDKit.SetFingerPosition and IDKit.GetFingerPosition in .NET or User.setFingerPosition and User.getFingerPosition in Java to change finger position.

3.5 Corresponding Irises

Two irises are considered to correspond to each other if they have the same iris position or if at least one of them has unknown iris position (see page 111).

Remarks

Only corresponding irises are compared during identification (see page 129) and verification (see page 137). You can use functions IEngine_SetIrisPosition (see page 47) and IEngine_GetIrisPosition (see page 47) in C++, IDKit.SetIrisPosition and IDKit.GetIrisPosition in .NET or User.setIrisPosition and User.getIrisPosition in Java to obtain and change iris position.

3.6 Custom Data String

Custom data string is arbitrary piece of data that applications can attach to every user record.

Remarks

Custom data string is a simple byte array. There is no restriction on size of the data stored in custom data string. IDKit SDK doesn't process custom data string in any way. Custom data string is intended to be used by application to store additional per-user data. It is stored as a BLOB in the database together with user record. It is similar to tags (see page 135), but it can store binary data (photo, documents) and it has unlimited size. Custom data string is not cached in memory. It is always loaded on demand from database.

Example Use Cases

It is up to the application developer to decide how to use the custom data string. There are several common scenarios:

- Interlink application database and IDKit SDK database by table key stored in custom data string.
- Store user identity (name, ID, etc.) or other fields by serializing them into XML document and storing this XML document in custom data string. XML encoding can be replaced by any other serialization mechanism.
- Store a list of successful identifications by this user, including time and other data.

API

- C++: IEngine_SetCustomData (see page 48), IEngine_GetCustomData (see page 49)
- .NET: IDKit.SetCustomData and IDKit.SetCustomData
- Java: User.setCustomData and User.getCustomData

3.7 External Database

External database is a biometric database that is not controlled by IDKit SDK.

Remarks

The integrated database support (see page 5) in IDKit SDK is optional. It is possible to perform identification (see page 129) and verification (see page 137) of user templates coming from external database. In order to match templates from external database, application loads all relevant templates into memory and calls identification or verification function that works with external database.

Following functions can work with templates from external database:

- IEngine_FindUserInMemory (see page 74), IEngine_FindFingerprintInMemory (see page 78), IEngine_MatchUsers (see page 87), IEngine_MatchFingerprints (see page 80), IEngine_MatchFaces (see page 83), IEngine_MatchIrises (see page 85) in C++
- BaseConnection.FindFingerprintInMemory, BaseConnection.FindUserInMemory, BaseConnection.MatchUsersEx, BaseConnection.MatchFingerprints, BaseConnection.MatchFaces, BaseConnection.MatchIrises in .NET
- IDKit.findFingerprintInMemory, IDKit.findUserInMemory, IDKit.matchUsersEx, IDKit.matchFingerprints, IDKit.matchFaces, IDKit.matchIrises in Java

In order to create or update the external database, application extracts templates (see page 136) for all added users, then exports template data using function IEngine_ExportUserTemplate (see page 56), IDKit.ExportUserTemplate in .NET or User.exportUserTemplate in Java, and saves template data in the external database.

Licensing (see page 130) works differently with external database. User limit is checked before every identification. If identification is performed over one part of the external database at a time, user limit applies to the whole database, not just the part submitted to the identification function.

3.8 FAR and Threshold

False Acceptance Rate (FAR) describes probability at which matching algorithm makes false acceptance errors.

Similarity scores returned by this library are normalized. More precisely, approximate relationship between similarity score and False Acceptance Rate (sometimes called also False Match Rate) is given by the following formula:

similarity score = -10 * log10 (FAR)

Please note that the normalization formula is only approximation. Score distribution can vary depending on various factors such as scanner surface area, finger placement, finger quality, finger positions (index fingers tend to give higher scores compared to little fingers, etc..)

Remarks

False Acceptance Rate (FAR) is one of several accuracy ratings that can be calculated for matching algorithm. False acceptance error is when matching algorithm identifies (see page 129) matching fingerprint in database for some probe fingerprint even though the two fingerprints do not belong to the same person.

FAR depends predominantly on the threshold (see page 137) selected. Increasing threshold reduces FAR, but it can aslo increase FRR (see page 129) (see Similarity Scores (see page 4)). FAR is influenced by quality of the database being searched and quality of the probe fingerprint.

3.8.1 Verification Threshold

For verification with a single finger, threshold corresponds to -10log(FAR). See the table below for corresponding values of threshold and given FAR depending on number of fingerprints per probe and enrolled user:

Probe Fingers	Enrolled Fingers	Threshold at FAR = 1%	Threshold at FAR = 0.1%	Threshold at FAR = 0.01%	Threshold at FAR = 0.001%	Threshold at FAR = 0.0001%
1	1	20	30	40	50	60
1	10	30	40	50	60	70
2	2	1	5	18	31	43
4	4	1	1	16	31	48
10	10	1	1	27	66	76

These values are just for verification, they can vary from data set with different quality or captured using different sensor. Default threshold is set to 40, which corresponds to FAR of 0.01% that can be also expressed as 1:10000 or 0.0001.

Parameters

The above was evaluated on our internal database using common single-finger scanner and with following parameters.

Rotation	60
Scanner	Futronic FS88
1-print	Right Index
2-prints	Right Index, Left Index
4-prints	Right Thumb, Left Thumb, Right Index, Left Index
10-prints	All 10 fingers

3.8.2 Identification Threshold

In identification the value of threshold can vary from given FAR, database size, number of fingerprints per probe user and number of fingerprints per gallery user. See the table below for values of thresholds recommended to use in identification (FAR of 0.01% can be also expressed as 1:10000 or 0.0001):

Probe Fingers	DB Size	Enrolled Fingers	Threshold at FAR = 1%	Threshold at FAR = 0.1%	Threshold at FAR = 0.01%	Threshold at FAR = 0.001%	Threshold at FAR = 0.0001%
1	100	1	35	47	59	72	87
1	1000	1	44	57	70	85	94
1	10000	1	52	66	80	93	128
1	100	10	42	54	67	80	88
1	1000	10	50	63	76	87	96
1	10000	10	59	73	86	99	128
2	100	2	1	12	24	39	46
2	1000	2	6	21	35	49	56
2	10000	2	14	29	44	61	102
4	100	4	1	10	28	43	60
4	1000	4	1	19	38	53	71
4	10000	4	9	28	47	67	82
10	100	10	1	21	56	87	114
10	1000	10	1	37	78	113	139
10	10000	10	9	54	94	127	142

Parameters

The above was evaluated on our internal database using common single-finger scanner and with following parameters.

Speed	4
Candidates Count	2
Rotation	60
Scanner	Futronic FS88
1-print	Right Index
2-prints	Right Index, Left Index
4-prints	Right Thumb, Left Thumb, Right Index, Left Index
10-prints	All 10 fingers

3.9 Biometric Template

Biometric template (or user template) is a data structure containing biometric information about one or more fingerprints, faces or irises.

Remarks

Templates are extracted (see page 136) from fingerprint images. IDKit SDK supports several template formats, see enumeration IENGINE_TEMPLATE_FORMAT (see page 109) in C++, IDKit.Enums.TemplateFormat in .NET or IDKit.TemplateFormat in Java. Depending on template format, biometric template can contain templates of multiple biometric modalities. Templates do not contain the original biometric images.

Compatibility

Innovatrics proprietary templates are backward compatible up to IDKit SDK version 7.1. Templates extracted using older IDKit SDK version could be imported (IEngine_ImportUserTemplate (see page 57) in C++, IDKit.ImportUserTemplate in .NET or User.importUserTemplate in Java) or loaded from database using newer IDKit version. This functionality has been removed in IDKit SDK version 7.2.0.

Up to IDKit SDK version 7.1 it was possible to export template downgraded to template version ICRS21 using parameter CFG_ICS_TEMPLATE_VERSION. ICRS21 template could be imported and used in IDKit SDK versions older than v2.60. This functionality has been removed in IDKit SDK version 7.2.0.

Since IDKit SDK version 7.2.0 only ICRS23 templates are supported.

3.9.1 Fingerprint Template

Fingerprint template is a data structure containing biometric information about one fingerprint.

Remarks

Templates are extracted (see page 136) from fingerprint images. IDKit SDK supports several template formats, see enumeration IENGINE_TEMPLATE_FORMAT (see page 109) in C++, IDKit.Enums.TemplateFormat in .NET or IDKit.TemplateFormat in Java. Depending on template format, fingerprint templates can contain a combination of minutiae and pattern data. Templates do not contain the original fingerprint image.

Compatibility

Innovatrics proprietary templates are backward compatible up to IDKit SDK version 7.1. Templates extracted using older IDKit SDK version could be imported (IEngine_ImportUserTemplate (see page 57) in C++, IDKit.ImportUserTemplate in .NET or User.importUserTemplate in Java) or loaded from database using newer IDKit version. This functionality has been removed in IDKit SDK version 7.2.0.

Up to IDKit SDK version 7.1 it was possible to export template downgraded to template version ICRS21 using parameter CFG_ICS_TEMPLATE_VERSION. ICRS21 template could be imported and used in IDKit SDK versions older than v2.60. This functionality has been removed in IDKit SDK version 7.2.0.

Since IDKit SDK version 7.2.0 only ICRS23 templates are supported.

3.9.2 Face Template

Face template is a data structure containing biometric information about one face. The size of one face template is 522 bytes.

Remarks

Templates are extracted (see page 136) from face images. IDKit SDK supports Innovatrics proprietary face template format only.Templates do not contain the original face image.

Compatibility

Face templates are not compatible across different template (extraction algorithm) version.

This incompatibility can lead to:

- cross-platform incompatibility
- incompatibility across IDKit Multi SDK versions

Cross-platform incompatibility

In IDKit Multi Android SDK only the FAST face template extraction mode is vailable.

The cross-platform template compatibility can be solved by setting the same face extraction algorithm on both platforms. This can be done by setting CFG_IFACE_EXTRACTION_MODE parameter in C++, ConnectionParameter.CfgIfaceExtractionMode in .NET and InstanceParameter.CFG_IFACE_EXTRACTION_MODE in Java. This functionality is present in IDKit Multi SDK since version 7.2.9.

Face template and IDKit Multi SDK versions compatibility

See below the table of face template compatibility and IDKit Multi SDK versions:

Face template version	Other compatible template versions	Face template extraction mode	Supported by IDKit Multi SDK (matching works)
1.5	1.7	Fast	4.5.x, 5.0.x, 5.1.x, 5.2.x
1.6	1.8	Accurate	4.5.x, 5.0.x, 5.1.x, 5.2.x
1.7	1.5	Fast	5.0.x, 5.1.x, 5.2.x
1.8	1.6	Accurate	5.0.x, 5.1.x, 5.2.x
1.9		Fast	5.1.x, 5.2.x
1.10		Accurate	5.1.x, 5.2.x
1.11		Accurate-server	5.1.x, 5.2.x
1.12		Accurate-server	5.3.x
1.14		Fast	6.0.x, 6.1.x, 6.3.x, 7.0.x, 7.1.x
1.15		Accurate	6.0.x, 6.1.x, 6.3.x, 7.0.x, 7.1.x
1.16		Accurate-server	6.0.x, 6.1.x, 6.3.x, 7.0.x, 7.1.x
1.17		Accurate-server	7.2.x, 7.3.x, 7.4.x
1.19		Fast	7.2.x, 7.3.x, 7.4.x, 7.5.x
1.20		Accurate	7.2.x, 7.3.x, 7.4.x, 7.5.x
1.22		Accurate-server	7.5.x, 7.6.x
1.23		Accurate-server-visa	7.6.x, 7.7.x
1.24		Fast	7.6.x

1.25		Balanced	7.6.x
1.26		Accurate	7.6.x
1.27		Accurate	7.7.x, 7.8.x
1.28		Accurate-server	7.7.x, 7.8.x
1.29		Balanced	7.7.x, 7.8.x
1.30		Fast	7.7.x, 7.8.x
1.31		Accurate-server-visa	7.8.x
1.32		Accurate-server-mask	8.0.x
1.33		Accurate	8.0.x
1.34		Accurate-server-wild	8.0.x
1.35		Balanced	8.0.x
1.36		Fast	8.0.x
1.37		Accurate-server-visa	8.0.x
1.38		Accurate-server-mask	8.0.x

3.9.3 Iris Template

Iris template is a data structure containing biometric information about one iris. The size of one iris template is 522 bytes.

Remarks

Templates are extracted (see page 136) from iris images. IDKit SDK supports Innovatrics proprietary iris template format only.Templates do not contain the original iris image.

Compatibility

Iris templates are not compatible across different template (extraction algorithm) versions.

Iris template and IDKit Multi SDK versions compatibility

See below the table of iris template compatibility and IDKit Multi SDK versions:

Iris template version	Supported by IDKit Multi SDK (matching works)
1.0	5.2.x, 5.3.x, 6.0.x, 6.1.x, 6.3.x, 7.0.x, 7.1.x
2.0	7.2.x, 7.3.x, 7.4.x, 7.5.x, 7.6.x, 7.7.x
3.0	7.8.x
3.1	8.0.x

3.10 FRR

False Rejection Rate (FRR) describes probability at which matching algorithm makes false rejection errors.

Remarks

False Rejection Rate (FRR) is one of several accuracy ratings that can be calculated for matching algorithm. False rejection error is when matching algorithm fails to identify (see page 129) a probe user (fingerprint, face or iris) in database even though the database contains user with biometric data (fingerprint, face or iris) of the same person.

FRR primarily depends on quality of the database being searched and quality of probe biometric data. FRR is somewhat influenced by threshold (see page 137). Decreasing threshold reduces FRR, but it can increase FAR (see page 122) significantly (see Similarity Scores (see page 4)).

3.11 Identification (one-to-many)

One-to-many (1:N) identification compares single probe user to a number of users in the database.

Remarks

Identification can be thought of as a sequence of 1:1 verifications (see page 137) executed in a loop against every user in the database, but identification is several orders of magnitude faster (see page 4) than a loop of verifications. Identification algorithm sorts database users by similarity score (see page 133) and returns one or more database users with the highest score. Use identification functions in C++ (see page 72), BaseConnection.Find* methods in .NET or IDKit.find* methods in Java to perform identification.

Identification algorithm is influenced by the following parameters:

- Similarity threshold (see page 137): CFG_SIMILARITY_THRESHOLD in C++, ConnectionParameter.CfgSimilarityThreshold in .NET and InstanceParameter.CFG_SIMILARITY_THRESHOLD in Java
- Maximum permitted rotation: CFG_MAX_ROTATION in C++, ConnectionParameter.CfgMaxRotation in .NET and InstanceParameter.CFG_MAX_ROTATION in Java
- Identification speed: CFG_IDENTIFICATION_SPEED in C++, ConnectionParameter.CfgIdentificationSpeed in .NET and InstanceParameter.CFG_IDENTIFICATION_SPEED in Java

Related Topics

One-to-One Verification (see page 137)

3.12 In-memory License

Buffer in memory that contains license (see page 130) data.

Remarks

In-memory license has the same format as file license, but it is stored in a memory buffer. It can be provided to IDKit SDK by calling:

- C++: IEngine_InitWithLicense (see page 8) instead of IEngine_InitModule (see page 8)
- .NET: IDKit.InitWithLicense instead of IDKit.InitModule
- Java: IDKit.initWithLicense instead of IDKit.initModule

In-memory license allows more flexible deployment of applications that integrate IDKit SDK. Applications using in-memory license can store the license in non-default locations, for example application-specific folders, system registry, or together with application license. In case the in-memory license provided to IDKit SDK is not valid for any reason, IDKit SDK continues searching for valid license in other locations (see page 130).

3.13 License

IDKit SDK checks digital license for purchased user limit.

Remarks

The digital license may be be stored in memory, in a license file, on USB dongle, or at Innovatrics license server (currently not used). License is not checked when connecting to ExpressID AFIS. The license contains information about purchased user limit for local database which can be obtained using IEngine_GetUserLimit (see page 98) in C++, IDKit.GetUserLimit in .NET or IDKit.getUserLimit in Java.

In case of external database (see page 121) and memory database, user limit applies to the size of the whole database, not just the part that is visible to IDKit SDK at any given moment. In case of multiple connections, user limit applies to the sum of sizes of all databases, not just to the size of the largest database.

IDKit SDK compares user limit to the current database size at several points. When user limit is reached, no further user registrations are allowed.

On Windows 7 and higher, IDKit SDK looks for the license in following order:

1. In-memory license (see page 130).
2. USB dongle.
3. User's private application data folder, usually C:\Users\<user>\AppData\Local\Innovatrics\iengine.lic.
4. System-wide application data folder, usually C:\ProgramData\Innovatrics\iengine.lic.
5. Current directory.

On Linux, IDKit SDK looks for the license in following order:

1. In-memory License (see page 130).
2. USB dongle.
3. $HOME/.innovatrics/iengine.lic.
4. /etc/innovatrics/iengine.lic.
5. $HOME/.idkit/iengine.lic.
6. /etc/idkit/iengine.lic.
7. Current directory.

On Android and Embedded system, it is recommended to initialize IDKit Embedded SDK, IDKit Android SDK or IDKit Multi Android SDK using In-memory License (see page 130).

3.14 Fingerprint Quality and Presence

Fingerprint quality as measured by Innovatrics template extraction algorithm. In addition to that, IDKit is able to measure also where fingerprint is present.

Remarks

Quality

Fingerprint quality is a value between 0 (lowest quality) and 100 (highest quality). It is calculated during template extraction (see page 136) and afterwards recorded in fingerprint template. It is linked to the total number of distinctive features found in the fingerprint image.

Fingerprint quality has some predictive power for identification errors (FAR (see page 122) and FRR (see page 129)). Higher quality fingerprints are more likely to result in higher recognition rate. Generally, quality above 40 is good enough for matching. Application should reject scanned fingerprints with quality below 40 during registration.

Application can submit low quality probe to identification (see page 129) function. IDKit SDK will try to perform identification with the available biometric information. Low quality probe fingerprints can increase FAR (see page 122) and they do rise FRR (see page 129) due to missing information. In order to control FAR (see page 122)and FRR (see page 129), it is recommended to check quality even for probe fingerprints. Quality above 30 should be sufficient for reliable identification.

Application can query fingerprint quality with function IEngine_GetFingerprintQuality (see page 51) in C++, IDKit.GetFingerprintQuality in .NET and User.getFingerprintQuality.

Presence

Fingerprint presence is additional quality indicator in IDKit SDK. It can have a value between 0 (lowest quality) and 100 (highest quality). Image quality number is calculated as an area where presence of fingerprint is detected. This function does not measure quality of the presented fingerprint pattern. It is calculated directly from fingerprint image, therefore presence evaluation it is very fast. Its primary use is intended for fingerprint presence detection and auto-capture trigger in real-time capture process.

Quality above 15 should be sufficient for reliable fingerprint placement detection, quality above 40 is sufficient for enrollment and identification.

Application can query fingerprint quality with function IEngine_GetFingerprintPresence (see page 50) in C++, IDKit.GetFingerprintPresence in .NET and User.getFingerprintPresence.

3.15 Face Quality

This is an Innovatrics-defined face quality. Face quality is a value ranging between 0 and 255 (0 - undefined, 1 - min quality, 255 - max quality) and it is stored in the header of face template. The face quality determines how relevant is the face template for matching and it relates to the quality of input face sample.

Application can query face quality with function IEngine_GetFaceQuality (see page 54) in C++, IDKit.GetFaceQuality in .NET and User.getFaceQuality.

3.16 Iris Quality

This is an Innovatrics-defined iris quality. Iris quality is a value ranging between 0 and 255 (0 - undefined, 1 - min quality, 255 - max quality) and it is stored in the header of iris template. The iris quality determines how relevant is the iris template for matching and it relates to the quality of input iris sample.

Application can query iris quality with function IEngine_GetIrisQuality (see page 54) in C++, IDKit.GetIrisQuality in .NET and User.getIrisQuality.

3.17 Score Consolidation

Score consolidation is a process of combining similarity scores (see page 133) of several corresponding fingerprint pairs (see page 120), face pairs and iris pairs into one common similarity score for whole user pair. It is capped at the maximal value of the similarity score (see page 133), which is 1000.

Remarks

Function used for similarity score (see page 133) consolidation of multiple corresponding fingerprint pairs,, face pairs and iris pairs cannot be expressed by simple formula. It is optimized for best possible accuracy for a given number of consolidated individual scores.

If user record contains multiple instances of same fingerprint position (e.g., user record contains three right index fingerprint templates (see page 126) in order to achieve better accuracy), all similarity scores of corresponding fingerprint (see page 120) pairs are put into set based on fingerprint position. Maximum similarity score is then evaluated for each fingerprint position set and consolidated score is calculated as a function of all those maximums.

If user record contains multiple biometric modalities (fingerprint, faces and irises) the consolidated score is calculated as a function of maximum similarity scores per modality and per fingerprint/iris position.

3.18 Selection

Selections specify database subsets where identification will be performed.

Remarks

If an application has to manage several databases, it is not necessary to build several AFIS systems. IDKit SDK can perform identification over application-specified subsets of one shared database. These subsets are simply arrays of user IDs. They

have purpose similar to tag queries (see page 133), but they are more flexible. For example, SQL query to an application database can be used to generate the list of IDs which is a more flexible mechanism than tag query. On the other hand, if selections are large (more than 1,000 IDs), they put noticeable load on the network when working with ExpressID AFIS.

API

- C++: IEngine_FindUserInSelection (see page 73), IEngine_FindFingerprintInSelection (see page 76)
- .NET: BaseConnection.FindUserInSelection, BaseConnection.FindFingerprintInSelection
- Java: IDKit.findUserInSelection, IDKit.findFingerprintInSelection

3.19 Similarity Score

Similarity score quantifies level of biometric similarity between two fingerprints, faces or irises.

Remarks

Similarity score is returned by identification (see page 129) and verification (see page 137) functions. It is a number in range 0 (lowest similarity) to 1000 (highest similarity). Score above similarity threshold (see page 137) indicates matching pair of fingerprints, faces or irises. For pure face identification and verification the highest possible similarity score is 100.

When a user pair has more than one corresponding fingerprint pair (see page 120), similarity scores of compared biometric pairs are automatically consolidated (see page 132) into one common similarity score for whole user pair.

When a user pair has more than one biometric modalities, similarity scores of compared biometric pairs are automatically consolidated (see page 132) into one common similarity score for whole user pair.

For more information: Similarity Scores (see page 4).

3.20 Tag Query

Tag query filters users by tag (see page 135) values.

Remarks

Tag query is a subset of SQL that defines queries over tag (see page 135) values and user IDs. Tag queries are evaluated with wire-level performance contrary to standard database queries. Tag query allows applications to narrow the set of users included in identification (see page 129). Queries can be also used alone to retrieve user IDs solely by tag values. Queries are similar to selections (see page 132), but they save network bandwidth in client-server environment when selecting large subsets of the database.

Examples Use Cases

Tag queries can be used in various ways:

- Binning, i.e. narrowing the set of users included in identification requests.
- To mark some user records as special. Markers can then be used to include/exclude groups of users in a tag query.
- To merge several partial AFIS systems into one global AFIS. Global AFIS makes it easier to perform queries that span several partial AFIS systems. It also improves load balancing for clustered systems. Thanks to tag query, data in partial AFIS systems can still be accessed when necessary.
- To retrieve a list of user IDs that have matching tag values or user IDs in certain range.

Supported SQL Subset

Following features of SQL are supported:

- All queries have the form SELECT USERID FROM TAG_CACHE WHERE *condition*.
- Table TAG_CACHE contains one row per user record. Columns include USERID and one column per tag name.
- Operators: AND, OR, NOT, =, <>, <, >, <=, >=, IS [NOT] NULL, [NOT] IN (constant list), (sub-condition).
- Operands: string constant, integer constant, column name (including USERID).
- Conversions between strings and 32-bit integers are performed automatically.
- Column has value NULL if the corresponding tag was not set on this user record.
- Unicode is not supported directly, but application can encode Unicode tag values in UTF-7.

Performance

Tag cache actually exists as a table in memory. It takes up space proportional to number of columns and number of rows. In ExpressID AFIS Government, tag cache rows are distributed over all nodes together with corresponding user records.

Query implementation is a brute-force filter that evaluates WHERE clause for every row in the tag cache. When used together with identification, tag query typically consumes less than 10 percent of total identification time. Tag query typically filters over 50 million rows per second per core (or 50,000 per millisecond per core). In ExpressID AFIS Government, evaluation of queries is evenly distributed to all nodes in the cluster and there is no network overhead, because tag cache records are co-located with template cache records.

API

- C++: IEngine_FindUserByQuery (see page 74), IEngine_FindFingerprintByQuery (see page 77), IEngine_GetUserIDsByQuery (see page 71)
- .NET: BaseConnection.FindFingerprintByQuery, BaseConnection.FindUserByQuery, BaseConnection.GetUserIDsByQuery
- Java: IDKit.findUserByQuery, IDKit.findFingerprintByQuery, IDKit.getUserIDsByQuery

Example

Select users with IDs in range from 3,000,001 to 4,000,000:

```
SELECT USERID FROM TAG_CACHE WHERE USERID >= 3000001 AND USERID <= 4000000
```

Select all users except one (useful when identified user is already registered):

```
SELECT USERID FROM TAG_CACHE WHERE USERID <> 1234567
```

Select all men, exclude all women:

```
SELECT USERID FROM TAG_CACHE WHERE GENDER = 'male'
```

Select users with age up to 30:

```
SELECT USERID FROM TAG_CACHE WHERE AGE <= 30
```

Select users born before 1.1.1980:

```
SELECT USERID FROM TAG_CACHE WHERE DATE_OF_BIRTH < 19800101
```

Select users by previously calculated and stored similarity score to their best match:

```
SELECT USERID FROM TAG_CACHE WHERE SCORE >= 13000
```

Select users from 3 cities:

```
SELECT USERID FROM TAG_CACHE WHERE TOWN IN ('CityA', 'CityB', 'CityC')
```

Select users with no local hometown (foreigners):

```
SELECT USERID FROM TAG_CACHE WHERE TOWN IS NULL
```

Select users with fingerprint quality below 40 (as recorded by the application during enrollment):

```
SELECT USERID FROM TAG_CACHE WHERE QUALITY < 40
```

3.21 Tags

User record can have persistent tags containing strings or numbers.

Remarks

Tags are a way to attach application-defined information to user records. They are similar to custom data string (see page 121), but they are more structured, they are cached in memory, they are stored in the database in an application-readable format, and they can be used in a tag query (see page 133).

Supported Types and Formats

Arbitrary number of tags can be attached to every user record. Every tag consists of a name and a value. Following restrictions apply:

- *Tag name* is any combination of letters, numbers, and underscores starting with a letter. Tag names are not case-sensitive. Tag names returned by IDKit SDK functions are always in uppercase form.
- *Tag value* is an arbitrary non-empty ASCII string.
- *Unicode* tag values are not supported directly, but application can encode Unicode strings in UTF-7 before storing them in tags.
- *Integers* are stored as strings, but tag query (see page 133) can compare them like integers.
- *Dates* can be encoded in a variety of ways. It is recommended to store dates as numbers in YYYYMMDD format. This format allows date comparisons to be done with integer operators.

Example Use Cases

Tags can be used in a variety of ways:

- To be used in tag queries. See tag query (see page 133) topic for use cases involving queries.
- To record various information about the user, for example ID card number, other application-specific IDs, or personal details (if they don't use Unicode characters).
- To record results of previous identifications. A de-duplication application might wish to store ID of the best match and its similarity score with every user record. It is then possible to use tag query to retrieve a list of users with matching score in certain range.

API

- C++: tag function group (see page 58)
- .NET: IDKit.GetIntTag, IDKit.SetIntTag, IDKit.GetStringTag, IDKit.SetStringTag, IDKit.GetTagName, IDKit.GetTagCount, IDKit.HasTag, IDKit.ClearTag
- Java: User.getTags method and User.UserTags class in Java

Database Structure

Tags are stored in table IENGINE_TAGS. Every row contains one name-value pair along with user ID. Names and values are ASCII strings that can be queried and manipulated with SQL queries. Note that changes to the table are not visible through IDKit SDK until the database is reopened and reloaded.

See tag query (see page 133) topic for information on how much memory is consumed by tag cache. Tags should be modified only trough Innovatrics API's. Manual modifications could harm the data in database.

3.22 Template Cache

Area of memory that contains biometric modality templates.

Remarks

IDKit SDK pre-loads all templates (see page 126) from database into memory in order to improve identification (see page 129) speed. Images and custom data string are not pre-loaded.

When IDKit SDK connects to ExpressID AFIS, all identification is performed remotely on the ExpressID AFIS and no pre-loading is needed at client side. Connections to memory database need no pre-loading, because all templates are already in memory.

3.23 Template Extraction

Template extraction is a process of converting image containing biometric data into a template, e. g. fingerprint template (see page 126), face template (see page 127) or iris template (see page 129) .

Remarks

Template extraction is a lengthy procedure (about 300ms for fingerprint), but it saves time during subsequent identification (see page 129) and verification (see page 137). It also saves memory and disk space, because templates are much smaller that images. Template extraction is performed in functions IEngine_AddFingerprint (see page 14), IEngine_AddFace (see page 22) or IEngine_AddIris (see page 34) in C++, IDKit.AddFaceFingerprint, IDKit.AddFace or IDKit.AddIris in .NET and User.addFingerprint, User.addFace or User.addIris in Java. Raw template data can be obtained using function IEngine_ExportUserTemplate (see page 56), IDKit.ExportUserTemplate in .NET and User.exportUserTemplate in Java.

3.24 Threads

Notes on using IDKit SDK in multi-threaded applications.

Remarks

IDKit SDK is **thread-safe**. API calls generally run in **parallel** when executed from multiple threads. Internal locks serialize API calls that require write access to the same resource (notably the same connection (see page 119)). Identification (see page 129) is **automatically parallelized** on multi-core processors. Template extraction (see page 136), the second most expensive operation in IDKit SDK, is not parallelized automatically, but application may execute several extractions in parallel.

In C++, there is one type of operation that is not thread-safe: deallocation of objects (IEngine_FreeUser (see page 13), IEngine_FreeCollection (see page 64)). Application is responsible for ensuring that the object is no longer used by any thread when its deallocation function is executed. In .NET and Java, this is done automatically.

Set environment variable OMP_NUM_THREADS to limit the number of threads of IEngine core used during identification, matching or extraction.

3.25 Verification (one-to-one)

One-to-one (1:1) verification compares two users, two fingerprints, two faces or two irises and calculates similarity score (see page 133) between the two.

Remarks

Fingerprint verification tries to find overlapping area between two fingerprints and compares features (minutiae and patterns) of the two fingerprints in this area. The resulting similarity score (see page 133) quantifies degree of similarity between the two fingerprints.

Face and iris verification compares features of the faces or irises and quantifies the degree of similarity using similarity score (see page 133).

Verification of two users is performed by comparing:

- each fingerprint of one user with each corresponding fingerprint (see page 120) of the other user,
- each face of one user with each face of the other user,
- each iris of one user with each corresponding iris (see page 121) of the other user,

Resulting set of partial similarity scores is consolidated (see page 132) into global user score.

Use verification functions in C++ (see page 79), BaseConnection.Match methods in .NET or IDKit.match methods in Java to perform verification.

Verification algorithm is influenced by the following parameters:

- Similarity threshold (see page 137): CFG_SIMILARITY_THRESHOLD in C++, ConnectionParameter.CfgSimilarityThreshold in .NET and InstanceParameter.CFG_SIMILARITY_THRESHOLD in Java
- Maximum permitted rotation: CFG_MAX_ROTATION in C++, ConnectionParameter.CfgMaxRotation in .NET and InstanceParameter.CFG_MAX_ROTATION in Java

Related Topics

One-to-Many Identification (see page 129)

3.26 Similarity Threshold

Similarity threshold defines minimum similarity score (see page 133) that is considered a match.

Remarks

While similarity score (see page 133) provides degree of certainty information that is in line with probabilistic algorithms of biometrics, most applications eventually have to decide whether particular similarity score means a match or not. Similarity threshold is used to divide the similarity spectrum into matching and non-matching range.

Threshold can be configured using CFG_SIMILARITY_THRESHOLD in C++, ConnectionParameter.CfgSimilarityThreshold in .NET and InstanceParameter.CFG_SIMILARITY_THRESHOLD in Java. Identification (see page 129) and verification (see page 137) functions only return matches above the current threshold.

Default threshold setting is suitable for most applications. You can adjust the threshold to control FAR (see page 122) and FRR (see page 129) (see section Similarity Scores (see page 4)). There is an approximate mapping between FAR and threshold (see page 122).

4 Symbol Reference

4.1 Macros

The following table lists macros in this documentation.

4.2 Types

The following table lists types in this documentation.

4.3 Structs, Records, Enums

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

5 Open source SW credits

Some of the components included in IDKit SDK are licensed under free or open source licenses. We wish to thank the contributors to those projects.

5.1 LibSVM

https://www.csie.ntu.edu.tw/~cjlin/libsvm/

```
Copyright (c) 2000-2014 Chih-Chung Chang and Chih-Jen Lin
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
3. Neither name of copyright holders nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.2 Caffe

http://caffe.berkeleyvision.org/

```
COPYRIGHT
All contributions by the University of California:
Copyright (c) 2014-2017 The Regents of the University of California (Regents)
All rights reserved.
All other contributions:
Copyright (c) 2014-2017, the respective contributors
All rights reserved.
Caffe uses a shared copyright model: each contributor holds copyright over
their contributions to Caffe. The project versioning records all such
contribution and copyright details. If a contributor wants to further mark
their specific copyright on a particular contribution, they should indicate
their copyright solely in the commit message of the change when it is
committed.
LICENSE
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
```

```
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
CONTRIBUTION AGREEMENT
By contributing to the BVLC/caffe repository through pull-request, comment,
or otherwise, the contributor releases their content to the
license and copyright terms herein.
```

5.3 OpenCV

http://opencv.org/

```
By downloading, copying, installing or using the software you agree to this license.
If you do not agree to this license, do not download, install,
copy or use the software.

License Agreement
For Open Source Computer Vision Library
(3-clause BSD License)

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:
  * Redistributions of source code must retain the above copyright notice,
    this list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright notice,
    this list of conditions and the following disclaimer in the documentation
    and/or other materials provided with the distribution.
  * Neither the names of the copyright holders nor the names of the contributors
    may be used to endorse or promote products derived from this software
    without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and
any express or implied warranties, including, but not limited to, the implied
warranties of merchantability and fitness for a particular purpose are disclaimed.
In no event shall copyright holders or contributors be liable for any direct,
indirect, incidental, special, exemplary, or consequential damages
(including, but not limited to, procurement of substitute goods or services;
loss of use, data, or profits; or business interruption) however caused
and on any theory of liability, whether in contract, strict liability,
or tort (including negligence or otherwise) arising in any way out of
the use of this software, even if advised of the possibility of such damage.
```

5.4 GTest

https://github.com/google/googletest

```
Copyright 2008, Google Inc.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
```

```
* Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
* Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.5 GFlags

https://github.com/gflags/gflags

```
Copyright (c) 2006, Google Inc.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
* Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.6 GLog

https://github.com/google/glog

```
Copyright (c) 2008, Google Inc.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
```

```
distribution.
* Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
A function gettimeofday in utilities.cc is based on
http://www.google.com/codesearch/p?hl=en#dR3YEbitojA/COPYING&q=GetSystemTimeAsFileTime%20lic
ense:bsd
The license of this code is:
Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi> and contributors
All Rights Reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
3. Neither the name(s) of the above-listed copyright holder(s) nor the
names of its contributors may be used to endorse or promote products
derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.7 OpenBlas

http://www.openblas.net/

```
Copyright (c) 2011-2014, The OpenBLAS Project
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in
the documentation and/or other materials provided with the
distribution.
3. Neither the name of the OpenBLAS project nor the names of
its contributors may be used to endorse or promote products
derived from this software without specific prior written
permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
```

```
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.8 Protobuf

https://developers.google.com/protocol-buffers/

```
This license applies to all parts of Protocol Buffers except the following:
- Atomicops support for generic gcc, located in
src/google/protobuf/stubs/atomicops_internals_generic_gcc.h.
This file is copyrighted by Red Hat Inc.
- Atomicops support for AIX/POWER, located in
src/google/protobuf/stubs/atomicops_internals_aix.h.
This file is copyrighted by Bloomberg Finance LP.
Copyright 2014, Google Inc.
All rights reserved.
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:
* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
* Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner
of the input file used when generating it. This code is not
standalone and requires a support library to be linked with it. This
support library is itself covered by the above license.
```

5.9 Global Matting

https://github.com/atilimcetin/global-matting

```
The MIT License (MIT)
Copyright (c) 2015 AtÄ±lÄ±m Ã‡etin
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

```
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

5.10 Boost

http://www.boost.org/

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

5.11 libpng

http://www.libpng.org/pub/png/src/libpng-LICENSE.txt

```
This copy of the libpng notices is provided for your convenience.  In case of
any discrepancy between this copy and the notices in the file png.h that is
included in the libpng distribution, the latter shall prevail.

COPYRIGHT NOTICE, DISCLAIMER, and LICENSE:

If you modify libpng you may insert additional notices immediately following
this sentence.

This code is released under the libpng license.

libpng versions 1.0.7, July 1, 2000 through 1.6.32, August 24, 2017 are
Copyright (c) 2000-2002, 2004, 2006-2017 Glenn Randers-Pehrson, are
derived from libpng-1.0.6, and are distributed according to the same
disclaimer and license as libpng-1.0.6 with the following individuals
added to the list of Contributing Authors:

   Simon-Pierre Cadieux
   Eric S. Raymond
   Mans Rullgard
```

```
   Cosmin Truta
   Gilles Vollant
   James Yu
   Mandar Sahastrabuddhe
   Google Inc.
   Vadim Barkov

and with the following additions to the disclaimer:

   There is no warranty against interference with your enjoyment of the
   library or against infringement.  There is no warranty that our
   efforts or the library will fulfill any of your particular purposes
   or needs.  This library is provided with all faults, and the entire
   risk of satisfactory quality, performance, accuracy, and effort is with
   the user.

Some files in the "contrib" directory and some configure-generated
files that are distributed with libpng have other copyright owners and
are released under other open source licenses.

libpng versions 0.97, January 1998, through 1.0.6, March 20, 2000, are
Copyright (c) 1998-2000 Glenn Randers-Pehrson, are derived from
libpng-0.96, and are distributed according to the same disclaimer and
license as libpng-0.96, with the following individuals added to the list
of Contributing Authors:

   Tom Lane
   Glenn Randers-Pehrson
   Willem van Schaik

libpng versions 0.89, June 1996, through 0.96, May 1997, are
Copyright (c) 1996-1997 Andreas Dilger, are derived from libpng-0.88,
and are distributed according to the same disclaimer and license as
libpng-0.88, with the following individuals added to the list of
Contributing Authors:

   John Bowler
   Kevin Bracey
   Sam Bushell
   Magnus Holmgren
   Greg Roelofs
   Tom Tanner

Some files in the "scripts" directory have other copyright owners
but are released under this license.

libpng versions 0.5, May 1995, through 0.88, January 1996, are
Copyright (c) 1995-1996 Guy Eric Schalnat, Group 42, Inc.

For the purposes of this copyright and license, "Contributing Authors"
is defined as the following set of individuals:

   Andreas Dilger
   Dave Martindale
   Guy Eric Schalnat
   Paul Schmidt
   Tim Wegner

The PNG Reference Library is supplied "AS IS".  The Contributing Authors
and Group 42, Inc. disclaim all warranties, expressed or implied,
including, without limitation, the warranties of merchantability and of
fitness for any purpose.  The Contributing Authors and Group 42, Inc.
assume no liability for direct, indirect, incidental, special, exemplary,
or consequential damages, which may result from the use of the PNG
Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this
source code, or portions hereof, for any purpose, without fee, subject
to the following restrictions:
```

```
  1. The origin of this source code must not be misrepresented.

  2. Altered versions must be plainly marked as such and must not
     be misrepresented as being the original source.

  3. This Copyright notice may not be removed or altered from any
     source or altered source distribution.

The Contributing Authors and Group 42, Inc. specifically permit, without
fee, and encourage the use of this source code as a component to
supporting the PNG file format in commercial products.  If you use this
source code in a product, acknowledgment is not required but would be
appreciated.

END OF COPYRIGHT NOTICE, DISCLAIMER, and LICENSE.

TRADEMARK:

The name "libpng" has not been registered by the Copyright owner
as a trademark in any jurisdiction.  However, because libpng has
been distributed and maintained world-wide, continually since 1995,
the Copyright owner claims "common-law trademark protection" in any
jurisdiction where common-law trademark is recognized.

OSI CERTIFICATION:

Libpng is OSI Certified Open Source Software.  OSI Certified Open Source is
a certification mark of the Open Source Initiative. OSI has not addressed
the additional disclaimers inserted at version 1.0.7.

EXPORT CONTROL:

The Copyright owner believes that the Export Control Classification
Number (ECCN) for libpng is EAR99, which means not subject to export
controls or International Traffic in Arms Regulations (ITAR) because
it is open source, publicly available software, that does not contain
any encryption software.  See the EAR, paragraphs 734.3(b)(3) and
734.7(b).

Glenn Randers-Pehrson
glennrp at users.sourceforge.net
April 1, 2017
```

5.12 zlib

```
zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.11, January 15th, 2017

Copyright (C) 1995-2017 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty.  In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
 claim that you wrote the original software. If you use this software
 in a product, an acknowledgment in the product documentation would be
 appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
 misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.
```

```
Jean-loup Gailly        Mark Adler
jloup@gzip.org          madler@alumni.caltech.edu
```

5.13 Jasper

https://github.com/mdadams/jasper/blob/master/LICENSE

```
JasPer License Version 2.0

Copyright (c) 2001-2016 Michael David Adams
Copyright (c) 1999-2000 Image Power, Inc.
Copyright (c) 1999-2000 The University of British Columbia

All rights reserved.

Permission is hereby granted, free of charge, to any person (the
"User") obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, and/or sell copies of the Software, and to permit
persons to whom the Software is furnished to do so, subject to the
following conditions:

1.  The above copyright notices and this permission notice (which
Includes the disclaimer below) shall be included in all copies or
substantial portions of the Software.

2.  The name of a copyright holder shall not be used to endorse or
promote products derived from the Software without specific prior
written permission.

THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS
LICENSE.  NO USE OF THE SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER
THIS DISCLAIMER.  THE SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS
"AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING
BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.  IN NO
EVENT SHALL THE COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL
INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING
FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION
WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.  NO ASSURANCES ARE
PROVIDED BY THE COPYRIGHT HOLDERS THAT THE SOFTWARE DOES NOT INFRINGE
THE PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS OF ANY OTHER ENTITY.
EACH COPYRIGHT HOLDER DISCLAIMS ANY LIABILITY TO THE USER FOR CLAIMS
BROUGHT BY ANY OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL
PROPERTY RIGHTS OR OTHERWISE.  AS A CONDITION TO EXERCISING THE RIGHTS
GRANTED HEREUNDER, EACH USER HEREBY ASSUMES SOLE RESPONSIBILITY TO SECURE
ANY OTHER INTELLECTUAL PROPERTY RIGHTS NEEDED, IF ANY.  THE SOFTWARE
IS NOT FAULT-TOLERANT AND IS NOT INTENDED FOR USE IN MISSION-CRITICAL
SYSTEMS, SUCH AS THOSE USED IN THE OPERATION OF NUCLEAR FACILITIES,
AIRCRAFT NAVIGATION OR COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL
SYSTEMS, DIRECT LIFE SUPPORT MACHINES, OR WEAPONS SYSTEMS, IN WHICH
THE FAILURE OF THE SOFTWARE OR SYSTEM COULD LEAD DIRECTLY TO DEATH,
PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE ("HIGH
RISK ACTIVITIES").  THE COPYRIGHT HOLDERS SPECIFICALLY DISCLAIM ANY
EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR HIGH RISK ACTIVITIES.
```

5.14 NBIS

https://www.nist.gov/services-resources/software/nist-biometric-image-software-nbis

```
This software was developed at the National Institute of Standards
and Technology (NIST) by employees of the Federal Government in the
course of their official duties. Pursuant to Title 17 Section 105
of the United States Code (link is external), this software is not
subject to copyright protection and is in the public domain. NIST
assumes no responsibility whatsoever for use by other parties of
its source code or open source server, and makes no guarantees,
expressed or implied, about its quality, reliability, or any other
characteristic.

This software has been determined to be outside the scope of the EAR
(see Part 734.3 of the EAR for exact details) as it has been created
solely by employees of the U.S. Government; it is freely distributed
with no licensing requirements; and it is considered public domain.
Therefore, it is permissible to distribute this software as a free
download from the internet.

Disclaimer
Specific software products identified here are used in order to
perform the code management tasks at hand. However, in no case does
identification of any commercial product, trade name, or vendor,
imply recommendation or endorsement by the National Institute of
Standards and Technology, nor does it imply that the products and
equipment identified are necessarily the best available for the
apurpose.
```

5.15 JPEG Group

https://spdx.org/licenses/IJG.html

```
Independent JPEG Group License

LEGAL ISSUES

In plain English:

    1. We don't promise that this software works. (But if you find any bugs, please let us
know!)
    2. You can use this software for whatever you want. You don't have to pay us.
    3. You may not pretend that you wrote this software. If you use it in a program, you
must acknowledge somewhere in your documentation that you've used the IJG code.

In legalese:

The authors make NO WARRANTY or representation, either express or implied, with respect to
this software, its quality, accuracy, merchantability, or fitness for a particular purpose.
This software is provided "AS IS", and you, its user, assume the entire risk as to its
quality and accuracy.

This software is copyright (C) 1991-1998, Thomas G. Lane. All Rights Reserved except as
specified below.

Permission is hereby granted to use, copy, modify, and distribute this software (or
portions thereof) for any purpose, without fee, subject to these conditions:

    (1) If any part of the source code for this software is distributed, then this README
file must be included, with this copyright and no-warranty notice unaltered; and any
additions, deletions, or changes to the original files must be clearly indicated in
accompanying documentation.
    (2) If only executable code is distributed, then the accompanying documentation must
state that "this software is based in part on the work of the Independent JPEG Group".
    (3) Permission for use of this software is granted only if the user accepts full
responsibility for any undesirable consequences; the authors accept NO LIABILITY for
damages of any kind.

These conditions apply to any software derived from or based on the IJG code, not just to
```

```
the unmodified library. If you use our work, you ought to acknowledge us.

Permission is NOT granted for the use of any IJG author's name or company name in
advertising or publicity relating to this software or products derived from it. This
software may be referred to only as "the Independent JPEG Group's software".

We specifically permit and encourage the use of this software as the basis of commercial
products, provided that all warranty or liability claims are assumed by the product vendor.

ansi2knr.c is included in this distribution by permission of L. Peter Deutsch, sole
proprietor of its copyright holder, Aladdin Enterprises of Menlo Park, CA. ansi2knr.c is
NOT covered by the above copyright and conditions, but instead by the usual distribution
terms of the Free Software Foundation; principally, that you must include source code if
you redistribute it. (See the file ansi2knr.c for full details.) However, since ansi2knr.c
is not needed as part of any program generated from the IJG code, this does not limit you
more than the foregoing paragraphs do.

The Unix configuration script "configure" was produced with GNU Autoconf. It is copyright
by the Free Software Foundation but is freely distributable. The same holds for its
supporting scripts (config.guess, config.sub, ltconfig, ltmain.sh). Another support script,
install-sh, is copyright by M.I.T. but is also freely distributable.

It appears that the arithmetic coding option of the JPEG spec is covered by patents owned
by IBM, AT&T, and Mitsubishi. Hence arithmetic coding cannot legally be used without
obtaining one or more licenses. For this reason, support for arithmetic coding has been
removed from the free JPEG software. (Since arithmetic coding provides only a marginal gain
over the unpatented Huffman mode, it is unlikely that very many implementations will
support it.) So far as we are aware, there are no patent restrictions on the remaining code.

The IJG distribution formerly included code to read and write GIF files. To avoid
entanglement with the Unisys LZW patent, GIF reading support has been removed altogether,
and the GIF writer has been simplified to produce "uncompressed GIFs". This technique does
not use the LZW algorithm; the resulting GIF files are larger than usual, but are readable
by all standard GIF decoders.

We are required to state that
"The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated.
GIF(sm) is a Service Mark property of CompuServe Incorporated."
```

5.16 SQLite3

https://sqlite.org/copyright.html

```
All of the code and documentation in SQLite has been dedicated to the public domain by the
authors. All code authors, and representatives of the companies they work for, have signed
affidavits dedicating their contributions to the public domain and originals of those
signed affidavits are stored in a firesafe at the main offices of Hwaci. Anyone is free to
copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either
in source code form or as a compiled binary, for any purpose, commercial or non-commercial,
and by any means.

The previous paragraph applies to the deliverable code and documentation in SQLite - those
parts of the SQLite library that you actually bundle and ship with a larger application.
Some scripts used as part of the build process (for example the "configure" scripts
generated by autoconf) might fall under other open-source licenses. Nothing from these
build scripts ever reaches the final deliverable SQLite library, however, and so the
licenses associated with those scripts should not be a factor in assessing your rights to
copy and use the SQLite library.

All of the deliverable code in SQLite has been written from scratch. No code has been taken
from other projects or from the open internet. Every line of code can be traced back to its
original author, and all of those authors have public domain dedications on file. So the
SQLite code base is clean and is uncontaminated with licensed code from other projects.
```

5.17 Android NDK

https://github.com/googlesamples/android-ndk/blob/master/LICENSE

```
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and
distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright
owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities
that control, are controlled by, or are under common control with that entity.
For the purposes of this definition, "control" means (i) the power, direct or
indirect, to cause the direction or management of such entity, whether by
contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising
permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including
but not limited to software source code, documentation source, and configuration
files.

"Object" form shall mean any form resulting from mechanical transformation or
translation of a Source form, including but not limited to compiled object code,
generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made
available under the License, as indicated by a copyright notice that is included
in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that
is based on (or derived from) the Work and for which the editorial revisions,
annotations, elaborations, or other modifications represent, as a whole, an
original work of authorship. For the purposes of this License, Derivative Works
shall not include works that remain separable from, or merely link (or bind by
name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version
of the Work and any modifications or additions to that Work or Derivative Works
thereof, that is intentionally submitted to Licensor for inclusion in the Work
by the copyright owner or by an individual or Legal Entity authorized to submit
on behalf of the copyright owner. For the purposes of this definition,
"submitted" means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems, and
issue tracking systems that are managed by, or on behalf of, the Licensor for
the purpose of discussing and improving the Work, but excluding communication
that is conspicuously marked or otherwise designated in writing by the copyright
owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf
of whom a Contribution has been received by Licensor and subsequently
incorporated within the Work.

2. Grant of Copyright License.
```

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and
You must cause any modified files to carry prominent notices stating that You changed the files; and
You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.
You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

```
7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the
Work (and each Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
including, without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are
solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise of
permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable law (such as deliberate
and grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special, incidental,
or consequential damages of any character arising as a result of this License or
out of the use or inability to use the Work (including but not limited to
damages for loss of goodwill, work stoppage, computer failure or malfunction, or
any and all other commercial damages or losses), even if such Contributor has
been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to
offer, and charge a fee for, acceptance of support, warranty, indemnity, or
other liability obligations and/or rights consistent with this License. However,
in accepting such obligations, You may act only on Your own behalf and on Your
sole responsibility, not on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.
```

5.18 JNA

https://github.com/java-native-access/jna/blob/master/LICENSE

```
Java Native Access project (JNA) is dual-licensed under 2
alternative Open Source/Free licenses: LGPL 2.1 or later and
Apache License 2.0. (starting with JNA version 4.0.0).

You can freely decide which license you want to apply to
the project.

You may obtain a copy of the LGPL License at:

http://www.gnu.org/licenses/licenses.html

A copy is also included in the downloadable source code package
containing JNA, in file "LGPL2.1", under the same directory
as this file.

You may obtain a copy of the Apache License at:

http://www.apache.org/licenses/

A copy is also included in the downloadable source code package
containing JNA, in file "AL2.0", under the same directory
as this file.
```

5.19 mbed TLS

https://tls.mbed.org/how-to-get

```
All the current versions of the mbed TLS library are distributed under the Apache 2.0
license and available from our Download area. In addition there are packaged versions of
the mbed TLS library that are distributed with the GNU Public License Version 2.0 (GPL
v2.0).

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and
distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright
owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities
that control, are controlled by, or are under common control with that entity.
For the purposes of this definition, "control" means (i) the power, direct or
indirect, to cause the direction or management of such entity, whether by
contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising
permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including
but not limited to software source code, documentation source, and configuration
files.

"Object" form shall mean any form resulting from mechanical transformation or
translation of a Source form, including but not limited to compiled object code,
generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made
available under the License, as indicated by a copyright notice that is included
in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that
is based on (or derived from) the Work and for which the editorial revisions,
annotations, elaborations, or other modifications represent, as a whole, an
original work of authorship. For the purposes of this License, Derivative Works
shall not include works that remain separable from, or merely link (or bind by
name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version
of the Work and any modifications or additions to that Work or Derivative Works
thereof, that is intentionally submitted to Licensor for inclusion in the Work
by the copyright owner or by an individual or Legal Entity authorized to submit
on behalf of the copyright owner. For the purposes of this definition,
"submitted" means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems, and
issue tracking systems that are managed by, or on behalf of, the Licensor for
the purpose of discussing and improving the Work, but excluding communication
that is conspicuously marked or otherwise designated in writing by the copyright
owner as "Not a Contribution."
```

```
"Contributor" shall mean Licensor and any individual or Legal Entity on behalf
of whom a Contribution has been received by Licensor and subsequently
incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the Work and such
Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby
grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free,
irrevocable (except as stated in this section) patent license to make, have
made, use, offer to sell, sell, import, and otherwise transfer the Work, where
such license applies only to those patent claims licensable by such Contributor
that are necessarily infringed by their Contribution(s) alone or by combination
of their Contribution(s) with the Work to which such Contribution(s) was
submitted. If You institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work or a
Contribution incorporated within the Work constitutes direct or contributory
patent infringement, then any patent licenses granted to You under this License
for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof
in any medium, with or without modifications, and in Source or Object form,
provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of
this License; and
You must cause any modified files to carry prominent notices stating that You
changed the files; and
You must retain, in the Source form of any Derivative Works that You distribute,
all copyright, patent, trademark, and attribution notices from the Source form
of the Work, excluding those notices that do not pertain to any part of the
Derivative Works; and
If the Work includes a "NOTICE" text file as part of its distribution, then any
Derivative Works that You distribute must include a readable copy of the
attribution notices contained within such NOTICE file, excluding those notices
that do not pertain to any part of the Derivative Works, in at least one of the
following places: within a NOTICE text file distributed as part of the
Derivative Works; within the Source form or documentation, if provided along
with the Derivative Works; or, within a display generated by the Derivative
Works, if and wherever such third-party notices normally appear. The contents of
the NOTICE file are for informational purposes only and do not modify the
License. You may add Your own attribution notices within Derivative Works that
You distribute, alongside or as an addendum to the NOTICE text from the Work,
provided that such additional attribution notices cannot be construed as
modifying the License.
You may add Your own copyright statement to Your modifications and may provide
additional or different license terms and conditions for use, reproduction, or
distribution of Your modifications, or for any such Derivative Works as a whole,
provided Your use, reproduction, and distribution of the Work otherwise complies
with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted
for inclusion in the Work by You to the Licensor shall be under the terms and
conditions of this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify the terms of
any separate license agreement you may have executed with Licensor regarding
such Contributions.

6. Trademarks.
```

```
This License does not grant permission to use the trade names, trademarks,
service marks, or product names of the Licensor, except as required for
reasonable and customary use in describing the origin of the Work and
reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the
Work (and each Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied,
including, without limitation, any warranties or conditions of TITLE,
NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are
solely responsible for determining the appropriateness of using or
redistributing the Work and assume any risks associated with Your exercise of
permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence),
contract, or otherwise, unless required by applicable law (such as deliberate
and grossly negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special, incidental,
or consequential damages of any character arising as a result of this License or
out of the use or inability to use the Work (including but not limited to
damages for loss of goodwill, work stoppage, computer failure or malfunction, or
any and all other commercial damages or losses), even if such Contributor has
been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to
offer, and charge a fee for, acceptance of support, warranty, indemnity, or
other liability obligations and/or rights consistent with this License. However,
in accepting such obligations, You may act only on Your own behalf and on Your
sole responsibility, not on behalf of any other Contributor, and only if You
agree to indemnify, defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason of your
accepting any such warranty or additional liability.
```

5.20 miniglog

https://github.com/tzutalin/miniglog/blob/master/LICENSE

```
Ceres Solver - A fast non-linear least squares minimizer
Copyright 2015 Google Inc. All rights reserved.
http://ceres-solver.org/

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be
  used to endorse or promote products derived from this software without
  specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
```

```
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

5.21 polly

https://github.com/ruslo/polly/blob/master/LICENSE

```
Copyright (c) 2013, Ruslan Baratov
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this
  list of conditions and the following disclaimer in the documentation and/or
  other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.22 OXFORD VGGFace2 Dataset

http://www.robots.ox.ac.uk/~vgg/data/vgg_face/licence.txt

```
The dataset consists of the crawled images of celebrities on the
Web. There are 2622 celebrities in the dataset and the dataset is
designed to have no overlap with the popular face recognition
benchmarks such as Labeled Faces in the Wild (LFW), YouTube Faces
Dataset and IARPA Janus Benchmark A (IJB-A).

Since the original images are subject to copyright, we do not make
them available directly. We instead provide URLs and associated face
detections from the dataset.

The images are covered under a Creative Commons
Attribution-NonCommercial 4.0 International license (Please read the
license terms here. http://creativecommons.org/licenses/by-nc/4.0/).

Downloading this dataset implies agreement to follow the same
conditions of non-commercial research for any modification and/or
re-distribution of the dataset in any form.


Additionally any entity using this dataset agrees to the following conditions:

THIS DATASET IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
```

```
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


Please cite [1] below if you make use of the dataset.


[1] O. M. Parkhi, A. Vedaldi, A. Zisserman
Deep Face Recognition
British Machine Vision Conference, 2015.
```

5.23 LevelDB

https://github.com/google/leveldb/blob/master/LICENSE

```
Copyright (c) 2011 The LevelDB Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

   * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
   * Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.24 Catch2

https://github.com/catchorg/Catch2/blob/master/LICENSE.txt

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:
```

```
The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

5.25 FakeIt

https://github.com/eranpeer/FakeIt/blob/master/LICENSE

```
The MIT License (MIT)

Copyright (c) 2013 Eran Pe'er

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

5.26 OpenSSL

https://www.openssl.org/source/license.html

```
 LICENSE ISSUES
  ==============

  The OpenSSL toolkit stays under a double license, i.e. both the conditions of
  the OpenSSL License and the original SSLeay license apply to the toolkit.
  See below for the actual license texts.

  OpenSSL License
  ---------------

/* ====================================================================
 * Copyright (c) 1998-2019 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
```

```
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

 Original SSLeay License
 -----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
```

```
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
```

5.27 bison

https://github.com/bincrafters/conan-bison/blob/stable/3.0.5/LICENSE.md

```
Copyright (c) 2017 Bincrafters

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), to deal in the Software
without restriction, including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons
to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

5.28 bzip2

https://github.com/bincrafters/conan-bison (see page 160)/blob/stable/3.0.5/LICENSE.md

```
This program, "bzip2", the associated library "libbzip2", and all
```

```
documentation, are copyright (C) 1996-2010 Julian R Seward.  All
rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. The origin of this software must not be misrepresented; you must
   not claim that you wrote the original software.  If you use this
   software in a product, an acknowledgment in the product
   documentation would be appreciated but is not required.

3. Altered source versions must be plainly marked as such, and must
   not be misrepresented as being the original software.

4. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior written
   permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, jseward@bzip.org
bzip2/libbzip2 version 1.0.6 of 6 September 2010
```

5.29 flex

https://github.com/westes/flex/blob/master/COPYING

```
Flex carries the copyright used for BSD software, slightly modified
because it originated at the Lawrence Berkeley (not Livermore!) Laboratory,
which operates under a contract with the Department of Energy:

Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006, 2007 The Flex Project.

Copyright (c) 1990, 1997 The Regents of the University of California.
All rights reserved.

This code is derived from software contributed to Berkeley by
Vern Paxson.

The United States Government has rights in this work pursuant
to contract no. DE-AC03-76SF00098 between the United States
Department of Energy and the University of California.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
```

```
Neither the name of the University nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.

This basically says "do whatever you please with this software except
remove this notice or take advantage of the University's (or the flex
authors') name".

Note that the "flex.skl" scanner skeleton carries no copyright notice.
You are free to do whatever you please with scanners generated using flex;
for them, you are not even bound by the above copyright.
```

5.30 gsl_microsoft

https://github.com/bincrafters/conan-gsl_microsoft/blob/stable/2.0.0/LICENSE.md

```
Copyright (c) 2017 Bincrafters

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), to deal in the Software
without restriction, including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons
to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

5.31 jaeger-client-cpp

https://github.com/jaegertracing/jaeger-client-cpp/blob/master/LICENSE

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
```

```
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      (except as stated in this section) patent license to make, have made,
      use, offer to sell, sell, import, and otherwise transfer the Work,
      where such license applies only to those patent claims licensable
      by such Contributor that are necessarily infringed by their
      Contribution(s) alone or by combination of their Contribution(s)
      with the Work to which such Contribution(s) was submitted. If You
      institute patent litigation against any entity (including a
      cross-claim or counterclaim in a lawsuit) alleging that the Work
      or a Contribution incorporated within the Work constitutes direct
      or contributory patent infringement, then any patent licenses
      granted to You under this License for that Work shall terminate
      as of the date such litigation is filed.
```

```
4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
```

```
      Work (including but not limited to damages for loss of goodwill,
      work stoppage, computer failure or malfunction, or any and all
      other commercial damages or losses), even if such Contributor
      has been advised of the possibility of such damages.

   9. Accepting Warranty or Additional Liability. While redistributing
      the Work or Derivative Works thereof, You may choose to offer,
      and charge a fee for, acceptance of support, warranty, indemnity,
      or other liability obligations and/or rights consistent with this
      License. However, in accepting such obligations, You may act only
      on Your own behalf and on Your sole responsibility, not on behalf
      of any other Contributor, and only if You agree to indemnify,
      defend, and hold each Contributor harmless for any liability
      incurred by, or claims asserted against, such Contributor by reason
      of your accepting any such warranty or additional liability.

   END OF TERMS AND CONDITIONS

   APPENDIX: How to apply the Apache License to your work.

      To apply the Apache License to your work, attach the following
      boilerplate notice, with the fields enclosed by brackets "{}"
      replaced with your own identifying information. (Don't include
      the brackets!)  The text should be enclosed in the appropriate
      comment syntax for the file format. We also recommend that a
      file or class name and description of purpose be included on the
      same "printed page" as the copyright notice for easier
      identification within third-party archives.

   Copyright {yyyy} {name of copyright owner}

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

5.32 jsonformoderncpp

https://github.com/vthiery/conan-jsonformoderncpp/blob/master/LICENSE

```
The MIT License (MIT)

Copyright (c) 2017 Vincent Thiery (vjmthiery@gmail.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

5.33 libevent

https://github.com/libevent/libevent/blob/master/LICENSE

```
Libevent is available for use under the following license, commonly known
as the 3-clause (or "modified") BSD license:

==============================
Copyright (c) 2000-2007 Niels Provos <provos@citi.umich.edu>
Copyright (c) 2007-2012 Niels Provos and Nick Mathewson

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
==============================

Portions of Libevent are based on works by others, also made available by
them under the three-clause BSD license above.  The copyright notices are
available in the corresponding source files; the license is as above.  Here's
a list:

log.c:
   Copyright (c) 2000 Dug Song <dugsong@monkey.org>
   Copyright (c) 1993 The Regents of the University of California.

strlcpy.c:
   Copyright (c) 1998 Todd C. Miller <Todd.Miller@courtesan.com>

win32select.c:
   Copyright (c) 2003 Michael A. Davis <mike@datanerds.net>

evport.c:
   Copyright (c) 2007 Sun Microsystems

ht-internal.h:
   Copyright (c) 2002 Christopher Clark

minheap-internal.h:
   Copyright (c) 2006 Maxim Yegorushkin <maxim.yegorushkin@gmail.com>

==============================

The arc4module is available under the following, sometimes called the
"OpenBSD" license:
```

```
   Copyright (c) 1996, David Mazieres <dm@uun.org>
   Copyright (c) 2008, Damien Miller <djm@openbsd.org>

   Permission to use, copy, modify, and distribute this software for any
   purpose with or without fee is hereby granted, provided that the above
   copyright notice and this permission notice appear in all copies.

   THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
   WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
   MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
   ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
   WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
   ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
   OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

==============================

The Windows timer code is based on code from libutp, which is
distributed under this license, sometimes called the "MIT" license.


Copyright (c) 2010 BitTorrent, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

5.34 opentracing-cpp

https://github.com/opentracing/opentracing-cpp/blob/master/LICENSE

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.
```

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

```
   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.
```

```
   9. Accepting Warranty or Additional Liability. While redistributing
      the Work or Derivative Works thereof, You may choose to offer,
      and charge a fee for, acceptance of support, warranty, indemnity,
      or other liability obligations and/or rights consistent with this
      License. However, in accepting such obligations, You may act only
      on Your own behalf and on Your sole responsibility, not on behalf
      of any other Contributor, and only if You agree to indemnify,
      defend, and hold each Contributor harmless for any liability
      incurred by, or claims asserted against, such Contributor by reason
      of your accepting any such warranty or additional liability.

   END OF TERMS AND CONDITIONS

   APPENDIX: How to apply the Apache License to your work.

      To apply the Apache License to your work, attach the following
      boilerplate notice, with the fields enclosed by brackets "{}"
      replaced with your own identifying information. (Don't include
      the brackets!)  The text should be enclosed in the appropriate
      comment syntax for the file format. We also recommend that a
      file or class name and description of purpose be included on the
      same "printed page" as the copyright notice for easier
      identification within third-party archives.

   Copyright The OpenTracing Authors

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

5.35 optional-lite

https://github.com/martinmoene/optional-lite/blob/master/LICENSE.txt

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

5.36 thrift

https://github.com/apache/thrift/blob/master/LICENSE

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."
```

```
   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
```

```
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

```
-----------------------------------------------
SOFTWARE DISTRIBUTED WITH THRIFT:

The Apache Thrift software includes a number of subcomponents with
separate copyright notices and license terms. Your use of the source
code for the these subcomponents is subject to the terms and
conditions of the following licenses.

-----------------------------------------------
Portions of the following files are licensed under the MIT License:

  lib/erl/src/Makefile.am

Please see doc/otp-base-license.txt for the full terms of this license.

-----------------------------------------------
For the aclocal/ax_boost_base.m4 and contrib/fb303/aclocal/ax_boost_base.m4 components:

#   Copyright (c) 2007 Thomas Porschberg <thomas@randspringer.de>
#
#   Copying and distribution of this file, with or without
#   modification, are permitted in any medium without royalty provided
#   the copyright notice and this notice are preserved.

-----------------------------------------------
For the lib/nodejs/lib/thrift/json_parse.js:

/*
    json_parse.js
    2015-05-02
    Public Domain.
    NO WARRANTY EXPRESSED OR IMPLIED. USE AT YOUR OWN RISK.

*/
(By Douglas Crockford <douglas@crockford.com>)
-----------------------------------------------
```

5.37 yaml-cpp

https://github.com/jbeder/yaml-cpp/blob/master/LICENSE

```
Copyright (c) 2008-2015 Jesse Beder.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

5.38 xsimd

https://github.com/xtensor-stack/xsimd/blob/master/LICENSE

```
Copyright (c) 2016, Johan Mabille, Sylvain Corlay,
         Wolf Vollprecht and Martin Renou
Copyright (c) 2016, QuantStack
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
OR BUSINESS INTERRUPTION) HOWEVERCAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY,  OR BUSINESS INTERRUPTION)
HOWEVEROR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

5.39 xercesc

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.
```

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly
negligent acts) or agreed to in writing, shall any Contributor be
liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

```
   9. Accepting Warranty or Additional Liability. While redistributing
      the Work or Derivative Works thereof, You may choose to offer,
      and charge a fee for, acceptance of support, warranty, indemnity,
      or other liability obligations and/or rights consistent with this
      License. However, in accepting such obligations, You may act only
      on Your own behalf and on Your sole responsibility, not on behalf
      of any other Contributor, and only if You agree to indemnify,
      defend, and hold each Contributor harmless for any liability
      incurred by, or claims asserted against, such Contributor by reason
      of your accepting any such warranty or additional liability.

   END OF TERMS AND CONDITIONS

   APPENDIX: How to apply the Apache License to your work.

      To apply the Apache License to your work, attach the following
      boilerplate notice, with the fields enclosed by brackets "[]"
      replaced with your own identifying information. (Don't include
      the brackets!)  The text should be enclosed in the appropriate
      comment syntax for the file format. We also recommend that a
      file or class name and description of purpose be included on the
      same "printed page" as the copyright notice for easier
      identification within third-party archives.

   Copyright [yyyy] [name of copyright owner]

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

5.40 openjpeg

https://github.com/bincrafters/conan-openjpeg/blob/testing/2.3.1/LICENSE.md

```
                    Copyright (c) 2017 - 2019 Bincrafters

   Permission is hereby granted, free of charge, to any person obtaining a copy
   of this software and associated documentation files (the "Software"), to deal
   in the Software without restriction, including without limitation the rights
   to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
   copies of the Software, and to permit persons to whom the Software is
   furnished to do so, subject to the following conditions:

   The above copyright notice and this permission notice shall be included in all
   copies or substantial portions of the Software.

   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
   IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
   FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
   AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
   LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
   OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
   SOFTWARE.
```

5.41 crc32c

https://github.com/google/crc32c/blob/master/LICENSE

```
                  Copyright 2017, The CRC32C Authors.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions are
   met:

      * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
      * Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following disclaimer
   in the documentation and/or other materials provided with the
   distribution.

      * Neither the name of Google Inc. nor the names of its
   contributors may be used to endorse or promote products derived from
   this software without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
   DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
   THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.42 snappy

https://github.com/google/snappy/blob/master/COPYING

```
                       Copyright 2011, Google Inc.
                           All rights reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions are
   met:

       * Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
       * Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following disclaimer
   in the documentation and/or other materials provided with the
   distribution.
       * Neither the name of Google Inc. nor the names of its
   contributors may be used to endorse or promote products derived from
   this software without specific prior written permission.

   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
   "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
   A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
   OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
   SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
   LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
```

```
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

5.43 libtiff

https://github.com/bincrafters/conan-libtiff/blob/testing/4.0.9/LICENSE.md

```
                  Copyright (c) 2017 - 2018 Bincrafters

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

5.44 icu

https://github.com/unicode-org/icu/blob/master/icu4c/LICENSE

```
COPYRIGHT AND PERMISSION NOTICE (ICU 58 and later)

Copyright Â© 1991-2020 Unicode, Inc. All rights reserved.
Distributed under the Terms of Use in https://www.unicode.org/copyright.html.

Permission is hereby granted, free of charge, to any person obtaining
a copy of the Unicode data files and any associated documentation
(the "Data Files") or Unicode software and any associated documentation
(the "Software") to deal in the Data Files or Software
without restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, and/or sell copies of
the Data Files or Software, and to permit persons to whom the Data Files
or Software are furnished to do so, provided that either
(a) this copyright and permission notice appear with all copies
of the Data Files or Software, or
(b) this copyright and permission notice appear in associated
Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT OF THIRD PARTY RIGHTS.
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS
NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL
DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,
DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THE DATA FILES OR SOFTWARE.
```

```
    Except as contained in this notice, the name of a copyright holder
    shall not be used in advertising or otherwise to promote the sale,
    use or other dealings in these Data Files or Software without prior
    written authorization of the copyright holder.
```

5.45 Intel MKL

https://software.intel.com/en-us/mkl https://software.intel.com/en-us/license/intel-simplified-software-license

Index

A

Add/Remove 14
Add/Remove Face 21
Add/Remove Fingerprint 14
Add/Remove Iris 33
Allocation 12
Android NDK 150

B

Biometric Template 124
Biometry 49
bison 160
Boost 144
Buffers (C++) 118
bzip2 160

C

C++ API Reference 7
Caffe 139
Catch2 157
Collections of Users 63
Connection String 119
Connections 9, 119
Corresponding Fingerprints 120
Corresponding Irises 121
crc32c 179
CRYPT_KEY_LENGTH macro 110
Custom Data String 121

D

Database 65
Database Connectivity 5

E

Error Codes 115
External Database 121

F

Face Properties 45
Face Quality 132
Face Template 127
FakeIt 158
FAR and Threshold 122
Fingerprint Properties 40
Fingerprint Quality and Presence 131
Fingerprint Template 126
flex 161
FRR 129
Function Quick Reference 113
Functions 7

G

GFlags 141
Global Matting 143
GLog 141
gsl_microsoft 162
GTest 140

I

icu 180
Identification 72
Identification (one-to-many) 129
Identification Speed 4
Identification Threshold 124
IEngine_AddFace function 22
IEngine_AddFaceFromFile function 26
IEngine_AddFaceFromFileRelative function 27
IEngine_AddFaceRAW function 24
IEngine_AddFaceRAWRelative function 25
IEngine_AddFaceRelative function 23
IEngine_AddFaceTemplate function 31
IEngine_AddFingerprint function 14
IEngine_AddFingerprintFromFile function 17
IEngine_AddFingerprintFromUser function 16
IEngine_AddFingerprintRAW function 15
IEngine_AddIris function 34
IEngine_AddIrisFromFile function 35

IEngine_AddIrisRAW function 35
IEngine_AddIrisTemplate function 38
IEngine_AttachFingerprintImage function 44
IEngine_ClearDatabase function 70
IEngine_ClearTag function 61
IEngine_ClearUser function 13
IEngine_CloseConnection function 11
IENGINE_COLLECTION type 103
IENGINE_CONFIG enumeration 103
IEngine_Connect function 10
IENGINE_CONNECTION type 107
IEngine_ConvertImage function 91
IEngine_ConvertImageToRaw function 90
IEngine_ConvertRawToImage function 89
IEngine_CopyUser function 12
IENGINE_CRITICAL_POINT structure 109
IEngine_DeserializeUser function 56
IEngine_ExportUserTemplate function 56
IENGINE_EXTRACTOR_ALGORITHM enumeration 111
IEngine_FindFingerprint function 75
IEngine_FindFingerprintByQuery function 77
IEngine_FindFingerprintInMemory function 78
IEngine_FindFingerprintInSelection function 76
IEngine_FindUser function 72
IEngine_FindUserByQuery function 74
IEngine_FindUserInMemory function 74
IEngine_FindUserInSelection function 73
IENGINE_FINGER_POSITION enumeration 107
IENGINE_FINGERPRINT_CLASS enumeration 110
IEngine_FingerprintImageExists function 42
IEngine_FreeCollection function 64
IEngine_FreeUser function 13
IEngine_GetAllUserIDs function 71
IEngine_GetCollectionIDs function 64
IEngine_GetCollectionSize function 64
IEngine_GetCustomData function 49
IEngine_GetDeltasAndCores function 51
IEngine_GetErrorMsg function 97
IEngine_GetFaceCount function 46
IEngine_GetFaceQuality function 54
IEngine_GetFaceTemplate function 92
IEngine_GetFingerPosition function 41
IEngine_GetFingerprintClass function 99
IEngine_GetFingerprintCount function 41
IEngine_GetFingerprintImage function 43
IEngine_GetFingerprintPresence function 50
IEngine_GetFingerprintPresenceRAW function 50
IEngine_GetFingerprintQuality function 51
IEngine_GetFingerprintResizeRatio function 99
IEngine_GetHardwareId function 96
IEngine_GetHardwareIdByMethod function 98
IEngine_GetIntermediateImages function 100
IEngine_GetIntTag function 59
IEngine_GetIrisCount function 46
IEngine_GetIrisPosition function 47
IEngine_GetIrisQuality function 54
IEngine_GetIrisTemplate function 100
IEngine_GetLicenseInformation function 96
IEngine_GetMemoryUsage function 97
IEngine_GetMinutiaeImage function 52
IEngine_GetMinutiaePoints function 101
IEngine_GetParameter function 93
IEngine_GetProductString function 96
IEngine_GetStringParameter function 94
IEngine_GetStringTag function 60
IEngine_GetTagCount function 62
IEngine_GetTagName function 62
IEngine_GetUser function 69
IEngine_GetUserCount function 70
IEngine_GetUserIDsByQuery function 71
IEngine_GetUserIfExists function 68
IEngine_GetUserLimit function 98
IENGINE_HARDWARE_ID_METHOD enumeration 112
IEngine_HasTag function 61
IENGINE_IMAGE_FORMAT enumeration 108
IEngine_ImportUserTemplate function 57
IEngine_InitCollection function 63
IEngine_InitConnection function 9
IEngine_InitModule function 8
IEngine_InitUser function 12
IEngine_InitWithLicense function 8
IENGINE_IRIS_IMAGE_TYPE type 111
IENGINE_IRIS_POSITION type 111
IEngine_MatchFace function 82

IEngine_MatchFaces function 83
IEngine_MatchFingerprint function 79
IEngine_MatchFingerprints function 80
IEngine_MatchFingerprints_transformation function 81
IENGINE_MATCHING_MODALITY enumeration 112
IEngine_MatchIris function 84
IEngine_MatchIrises function 85
IEngine_MatchUser function 86
IEngine_MatchUsers function 87
IEngine_MatchUsersModalities function 88
IEngine_MatchUsersModalitiesWithIntermediates function 101
IEngine_MatchUsersWithIntermediates function 102
IENGINE_MINUTIAE structure 108
IEngine_RegisterUser function 65
IEngine_RegisterUserAs function 66
IEngine_RemoveFace function 33
IEngine_RemoveFingerprint function 21
IEngine_RemoveIris function 40
IEngine_RemoveUser function 69
IEngine_SaveFingerprintImage function 45
IEngine_SaveMinutiaeImage function 53
IEngine_SelectConnection function 10
IEngine_SerializeUser function 55
IEngine_SetCryptKey function 95
IEngine_SetCustomData function 48
IEngine_SetFace function 27
IEngine_SetFaceRAW function 29
IEngine_SetFaceRAWRelative function 30
IEngine_SetFaceRelative function 28
IEngine_SetFaceTemplate function 32
IEngine_SetFingerPosition function 42
IEngine_SetFingerprint function 18
IEngine_SetFingerprintFromFile function 20
IEngine_SetFingerprintFromUser function 19
IEngine_SetFingerprintRAW function 18
IEngine_SetIntTag function 58
IEngine_SetIris function 36
IEngine_SetIrisPosition function 47
IEngine_SetIrisRAW function 37
IEngine_SetIrisTemplate function 39
IEngine_SetLogFile function 95
IEngine_SetParameter function 93
IEngine_SetStringParameter function 94
IEngine_SetStringTag function 59
IEngine_SetTracingSpanContextJson function 102
IENGINE_TEMPLATE_FORMAT enumeration 109
IEngine_TerminateModule function 9
IEngine_UpdateUser function 67
IENGINE_USER type 110
IEngine_UserExists function 68
Image Processing 6
Images 89
Import/Export 55
Initialization 8
In-memory License 130
Intel MKL 181
Iris Properties 46
Iris Quality 132
Iris Template 129

J

jaeger-client-cpp 162
Jasper 147
JNA 152
JPEG Group 148
jsonformoderncpp 165

L

LevelDB 157
libevent 166
libpng 144
LibSVM 139
libtiff 180
License 130

M

MAX_CRITICAL_POINTS_COUNT macro 110
mbed TLS 153
miniglog 155

N

NBIS 147
Notes 118

O

Open source SW credits 139
OpenBlas 142
OpenCV 140
openjpeg 178
OpenSSL 158
opentracing-cpp 167
optional-lite 170
Overview 1
OXFORD VGGFace2 Dataset 156

P

polly 156
Product Differences 2
Properties 40
Protobuf 143

S

Score Consolidation 132
Selection 132
Settings and Constants 92
Similarity Score 133
Similarity Scores 4
Similarity Threshold 137
snappy 179
SQLite3 149
Supported Platforms 3

T

Tag Query 133
Tags 58, 135
Template Cache 136
Template Extraction 136
Threads 136
thrift 171
Types, Structures, Enumerations 102

U

Usage Overview 1
User Record 11

V

Verification 79
Verification (one-to-one) 137
Verification Threshold 123

X

xercesc 175
xsimd 175

Y

yaml-cpp 174

Z

zlib 146