Innovatrics ANSI/ISO Generator &Matcher

Version 3.2.1.0

Table of Contents

1 Overview

The Innovatrics ANSI/ISO Generator & Matcher is designed to

- **generate ANSI/INCITS 378 and ISO/IEC 19794-2 compliant templates**
- **match ANSI/INCITS 378 and ISO/IEC 19794-2 compliant templates**

Features

- Innovatrics ANSI/ISO Generator (extractor) takes as input raw fingerprint images and transforms them to ANSI/ISO compliant templates.
- Innovatrics ANSI/ISO Matcher takes as input two ANSI/ISO compliant templates and returns corresponding similarity score.
- Image re-sizing feature for rescaling the input image to 500 DPI image
- Image conversion (WSQ, BMP, PNG, JPEG2K), possibility to set bit rate for WSQ and JPEG2K formats
- Template conversion (ANSI INCITS 378, ISO/IEC 19794-2:2005, ISO/IEC 19794-2:2011, ISO/IEC 19794-2:2005 compact card, ILO SID)
- Template data manipulation functionality for detailed analysis of fingerprint ANSI/ISO template
- Template merging for combining multiple finger views into one ANSI/ISO template
- The API of the library is based on MINEX API specifications designed by NIST.
- All functions from this library are thread-safe.

Supported platforms

Innovatrics ANSI/ISO SDK is officially supported for following OS and architectures

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

Architecture	NDK	API	Android OS
armv7a, x86	r15c	14 and higher	4 and higher
arm64-v8a, x86_64	r15c	21 and higher	5 and higher

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

.NET and Java connector

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

2 Fingerprint Image Data

The Generator takes as input fingerprint supplied to the SDK in uncompressed raw 8-bit (one byte per pixel) grayscale format. The recorded order of the scan image is illustrated in figure 1. The origin is the upper left corner of the image. The x-coordinate (horizontal) position shall increase positively from the origin to the right side of the image. The y-coordinate (vertical) position shall increase positively from the origin to the bottom of the image.

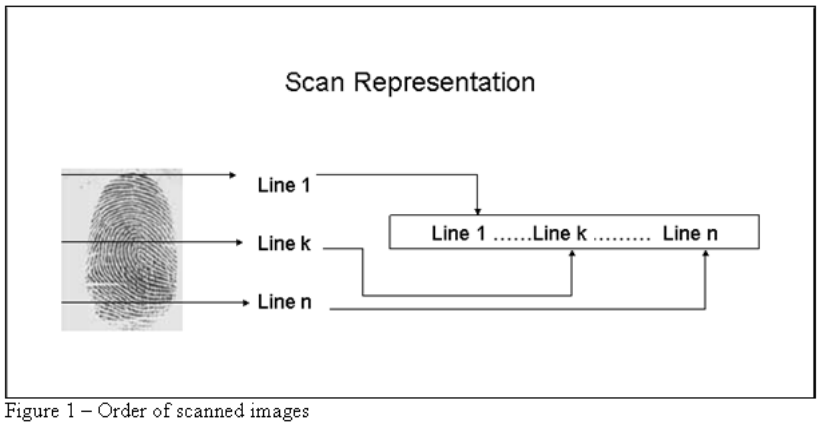

Figure 1 – Order of scanned images

Supported Image Formats

Supported image formats are enumerated in IENGINE_IMAGE_FORMAT (see page 27). In addition, ANSI/ISO SDK supports RAW image format through functions IEngine_ConvertToRaw (see page 11) and IEngine_ConvertRawToImage (see page 10).

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.

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

BMP Support

BMP image format can be found in multiple variants. In ANSI/ISO 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.

Notes

The resolution of images supplied to the Generator should be 500 DPI. ANSI&ISO SDK provides functionality for rescaling the input image.

3 Similarity Scores

Similarity scores returned by matching functions (ANSI_VerifyMatch (see page 19), ANSI_VerifyMatchEx (see page 19), ISO_VerifyMatch, ISO_VerifyMatchEx) are integer values ranging from 0 to 1000. Higher score means higher similarity of compared templates, lower score means lower similarity. When deciding whether two fingerprint templates are from the same finger (matching fingers), one has to compare matching score to a similarity threshold. Threshold (see page 31) defines False Accept Rate (FAR) and False Reject Rate (FRR) of the application. Higher threshold means higher FRR and lower FAR, lower threshold means lower FRR but higher FAR. Recommended setting for a standard biometric application is a similarity threshold corresponding to False Accept Rate from 1:1000 to 1:10000.

Notes

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..)

Example

Given the above normalization formula, threshold for FAR 1:1000 is approximately 30, threshold for FAR 1:10000 is approximately 40. For more information refer to section Threshold (see page 31).

4 Library Functions

4.1 Init, Terminate and other General Functions

Functions

	Name	Description
	IEngine_Init (see page 4)	Initializes the library
	IEngine_SetLicenseContent (see page 4)	Activates the library with a license key
	IEngine_Terminate (see page 5)	Terminates the use of the library
	IEngine_GetHwid (see page 5)	Returns hardware id of the device
	IEngine_GetHwidByMethod (see page 6)	Returns hardware id of the device based on selected method
	IEngine_GetVersion (see page 6)	Returns the library version
	IEngine_GetVersionString (see page 6)	Returns the library version as a string
	IEngine_GetImageQuality (see page 7)	Returns quality of a fingerprint image

4.1.1 IEngine_Init Function

Initializes the library

C++

```
IENGINE_API int IEngine_Init();
```

Description

This function initializes and checks the integrity of the library and verifies the validity of the license. It should be called prior to any other function from the library.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
50000	Errors greater and equal to 50000 are license related errors. Please use IEngine_GetErrorMessage to get detailed information about License Error.

Remarks

Function is not thread-safe. Protect parallel invocation of the function with a mutex.

Related Topics

IEngine_Terminate (see page 5), IEngine_GetVersion (see page 6)

4.1.2 IEngine_SetLicenseContent Function

Activates the library with a license key

C++

```
IENGINE_API int IEngine_SetLicenseContent(const unsigned char * licenseContent, int length);
```

Description

Use this function if you want to avoid storing license files on filesystem. This prevents a potential hacker to steal your license file information. This function has to be called before function IEngine_Init (see page 4).

Parameters

`const unsigned char * licenseContent`

[in] License information as provided by Innovatrics

`int length`

[in] Total length of license data

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
50000	Errors greater and equal to 50000 are license related errors. Please use IEngine_GetErrorMessage to get detailed information about License Error

Remarks

Function is not thread-safe. Protect parallel invocation of the function with a mutex.

4.1.3 IEngine_Terminate Function

Terminates the use of the library

C++

```
IENGINE_API int IEngine_Terminate();
```

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.

Remarks

Function is not thread-safe. Protect parallel invocation of the function with a mutex.

Related Topics

IEngine_Init (see page 4)

4.1.4 IEngine_GetHwid Function

Returns hardware id of the device

C++

```
IENGINE_API int IEngine_GetHwid(char * hwid, int * length);
```

Parameters

`char * hwid`

[out] Pointer to memory where hwid string will be stored.

`int * length`

[in/out] On input, specifies length of hwid array, on output, specifies required length needed to fill in hardware id to hwid array.

Notes

This function is deprecated and will likely be removed in a future version of the library.

4.1.5 IEngine_GetHwidByMethod Function

Returns hardware id of the device based on selected method

C++

```
IENGINE_API int IEngine_GetHwidByMethod(IENGINE_HWID_METHOD method, char * hwid, int *
length);
```

Parameters

`IENGINE_HWID_METHOD method`

[in] Method of obtaining the hardware id

`char * hwid`

[out] Pointer to memory where hwid string will be stored.

`int * length`

[in/out] On input, specifies length of hwid array, on output, specifies required length needed to fill in hardware id to hwid array.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_NOTSUPPORTED	The function is not supported for this version of product.
50000	Errors greater and equal to 50000 are license related errors. Please use IEngine_GetErrorMessage to get detailed information about License Error

Remarks

Function is not thread-safe. Protect parallel invocation of the function with a mutex.

4.1.6 IEngine_GetVersion Function

Returns the library version

C++

```
IENGINE_API int IEngine_GetVersion(IENGINE_VERSION * version);
```

Description

DEPRECATED: please prefer to use IEngine_GetVersionString (see page 6). This function returns the version number of the library

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Version

[out] Pointer to the structure where the library version will be stored

Related Topics

IEngine_GetVersionString (see page 6)

4.1.7 IEngine_GetVersionString Function

Returns the library version as a string

C++

```
IENGINE_API const char * IEngine_GetVersionString();
```

Description

This function returns the version string of the library

Returns

Version string

4.1.8 IEngine_GetImageQuality Function

Returns quality of a fingerprint image

C++

```
IENGINE_API int IEngine_GetImageQuality(int width, int height, const BYTE * rawImage, int * quality);
```

Description

This function returns quality of the input fingerprint image. Image quality number is calculated in accordance with the general guidelines contained in Section 7.2.5 of ANSI/INCITS 381 standard (Section 8.3.7.3 Quality score of ISO/IEC 19794-4 international standard).

Parameters

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

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

`const BYTE * rawImage`
[in] Pointer to the uncompressed raw image for template creation

`int * quality`
[out] Fingerprint image quality, the output range is from 0 (lowest quality) to 100 (highest quality)

4.2 Conversion Functions

Functions

	Name	Description
	IEngine_LoadBMP (see page 8)	Loads bmp image from file and converts it to raw 8-bit format
	IEngine_ConvertBMP (see page 8)	Reads bmp image from memory and converts it to raw 8-bit format
	IEngine_MakeBMP (see page 9)	Converts raw image into grayscale bmp image
	IEngine_ConvertRawToImage (see page 10)	Converts raw 8-bit image into formatted image
	IEngine_ConvertToRaw (see page 11)	Reads formatted image from memory and converts it to raw 8-bit format
	IEngine_ConvertIso19794_4ToRaw (see page 12)	Converts ISO 19794-4:2011 (FIR version 2.0) format into raw image format.
	IEngine_ConvertRawToIso19794_4 (see page 12)	Converts raw image into ISO 19794-4:2011 (FIR version 2.0) format.
	IEngine_ResizeImage (see page 13)	Reads raw 8-bit image from memory and creates new 8-bit 500dpi raw image.
	IEngine_ResizeImageInPlace (see page 14)	Reads raw 8-bit image from memory and resizes it to 500dpi in the same memory space.
	ANSI_ConvertToISO (see page 15)	Converts ANSI/INCITS 378 compliant template to ISO/IEC 19794-2 compliant template

4.2.1 IEngine_LoadBMP Function

Loads bmp image from file and converts it to raw 8-bit format

C++

```
IENGINE_API int IEngine_LoadBMP(const char * filename, int * width, int * height, BYTE * rawImage, int * length);
```

Description

This function reads bmp image contained in a file and converts it to raw 8-bit format as described in paragraph Fingerprint Image Data (see page 2)

Parameters

`const char * filename`

[in] Name of the file containing the input bmp image

`int * width`

[out] On return, contains the width of converted image

`int * height`

[out] On return, contains the height of converted image

`BYTE * rawImage`

[out] Pointer to memory space where raw image will be written

`int * length`

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

Returns

If rawImage is NULL or if length parameter is less then the size of the input image after conversion to 8-bit raw format, length parameter will be set to the required length of rawImage array.

If rawImage is not NULL and if length is greater or equal to the total memory size required to store the input image in 8-bit raw format, input bmp image will be converted to 8-bit raw image format and written into memory space pointed by rawImage parameter.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_FILE	Error occurred while opening/reading file.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Related Topics

IEngine_ConvertBMP (see page 8), IEngine_MakeBMP (see page 9)

4.2.2 IEngine_ConvertBMP Function

Reads bmp image from memory and converts it to raw 8-bit format

C++

```
IENGINE_API int IEngine_ConvertBMP(const BYTE * bmpImage, int * width, int * height, BYTE * rawImage, int * length);
```

Description

This function reads bmp image encoded in a byte array and converts it to raw 8-bit format as described in paragraph

Fingerprint Image Data (see page 2)

Parameters

`const BYTE * bmpImage`

[in] Pointer to the image in BMP format stored in the memory

`int * width`

[out] On return, contains the width of converted image

`int * height`

[out] On return, contains the height of converted image

`BYTE * rawImage`

[out] Pointer to memory space where raw image will be written

`int * length`

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

Returns

If rawImage is NULL or if length parameter is less then the size of the input image after conversion to 8-bit raw format, length parameter will be set to the required length of rawImage array.

If rawImage is not NULL and if length is greater or equal to the total memory size required to store the input image in 8-bit raw format, input bmp image will be converted to 8-bit raw image format and written into memory space pointed by rawImage parameter.

Return Values

Return Values	Description
`IENGINE_E_NOERROR`	No error occurred.
`IENGINE_E_BADFORMAT`	Unsupported image format.
`IENGINE_E_FILE`	Error occurred while opening/reading file.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

Related Topics

IEngine_LoadBMP (see page 8), IEngine_MakeBMP (see page 9)

4.2.3 IEngine_MakeBMP Function

Converts raw image into grayscale bmp image

C++

```
IENGINE_API int IEngine_MakeBMP(int width, int height, const BYTE * rawImage, BYTE *
bmpImageData, int * length);
```

Description

This function converts 8-bit raw image with specified dimensions into corresponding bmp grayscale image. The input raw 8-bit format is described in paragraph Fingerprint Image Data (see page 2)

Parameters

`int width`

[in] Width of the input raw image

`int height`

[in] Height of the input raw image

`const BYTE * rawImage`

[in] Pointer to raw image data

`BYTE * bmpImageData`

[out] Pointer to memory space where bmp image will be stored

`int * length`

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

equal to the total size of the raw image after conversion to bmp format.

Returns

If bmpImageData is NULL or if length parameter is less then the size of the input image after conversion to bmp format, length parameter will be set to the required length of bmpImageData array.

If bmpImageData is not NULL and if length is greater or equal to the total memory size required to store the input image in bmp format, input raw image will be converted to bmp image and written into memory space pointed by bmpImageData parameter.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_FILE	Error occurred while opening/reading file.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Related Topics

IEngine_ConvertBMP (see page 8), IEngine_LoadBMP (see page 8)

4.2.4 IEngine_ConvertRawToImage Function

Converts raw 8-bit image into formatted image

C++

```
IENGINE_API int IEngine_ConvertRawToImage(const BYTE * rawImage, int width, int height,
BYTE * outImage, IENGINE_IMAGE_FORMAT imageFormat, int rate, int * length);
```

Description

This function reads raw 8-bit image and encodes it into specified image format

Parameters

`const BYTE * rawImage`

[in] Pointer to the uncompressed raw image

`int width`

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

`int height`

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

`BYTE * outImage`

[out] Pointer to memory space where output (converted) image will be written

`IENGINE_IMAGE_FORMAT imageFormat`

[in] Indicates image format in which the output image should be encoded

`int rate`

[in] Specifies bitrate for JPEG2000 and WSQ images. Ignored for other image formats. Value of this parameter for JPEG2000 format is transformed to maximum bitrate per pixel, using formula 1/rate. (value 20 means bitrate of maximum 1:20 bits per pixel). Valid range of parameter values for WSQ format is [1, INT_MAX]. The meaningful range of parameter values for WSQ format is [1, 10]. To perform a correct conversion, the DPI resolution of the converting image must be 500.

`int * length`

[in/out] On input, this value is interpreted as the total size of allocated memory pointed by outImage parameter. On return, this parameter will be equal to the total size of the output image after conversion to the specified format.

Returns

If outImage is NULL or if length parameter is less than the size of the ouput image, length parameter will be set to the required length of outImage array.

If outImage is not NULL and if length parameter is greater or equal to the total memory size required to store the output image, input image will be encoded as specified and written into memory space pointed by outImage parameter.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_BADDPI	Image does not have required DPI or bitrate.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

4.2.5 IEngine_ConvertToRaw Function

Reads formatted image from memory and converts it to raw 8-bit format

C++

```
IENGINE_API int IEngine_ConvertToRaw(const BYTE * imageData, int imageLength,
IENGINE_IMAGE_FORMAT imageFormat, int * width, int * height, BYTE * rawImage, int *
rawImageLength);
```

Description

This function reads image encoded in a byte array and converts it to raw 8-bit format as described in paragraph Fingerprint Image Data (see page 2)

Parameters

`const BYTE * imageData`

[in] Pointer to the image in BMP format stored in the memory

`int imageLength`

[in] Indicates lenght of input image

`IENGINE_IMAGE_FORMAT imageFormat`

[in] Indicates format in which input image is encoded

`int * width`

[out] On return, contains the width of converted image

`int * height`

[out] On return, contains the height of converted image

`BYTE * rawImage`

[out] Pointer to memory space where raw image will be written

`int * rawImageLength`

[in/out] On input, rawImageLength value is interpreted as the total size of allocated memory pointed by rawImage parameter. On return, this parameter will be equal to the total size of the image after conversion to raw format.

Returns

If rawImage is NULL or if rawImageLength parameter is less than the size of the input image after conversion to 8-bit raw format, rawImageLength parameter will be set to the required length of rawImage array.

If rawImage is not NULL and if rawImageLength is greater or equal to the total memory size required to store the input image in 8-bit raw format, input image will be converted to 8-bit raw image format and written into memory space pointed by rawImage parameter.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

4.2.6 IEngine_ConvertIso19794_4ToRaw Function

Converts ISO 19794-4:2011 (FIR version 2.0) format into raw image format.

C++

```
IENGINE_API int IEngine_ConvertIso19794_4ToRaw(const unsigned char * isoFingerImage,
unsigned int isoImageLength, int * width, int * height, unsigned char * fingerPosition,
unsigned char * imageFormat, unsigned int * dpiX, unsigned int * dpiY, unsigned char *
rawImage, int * rawImageLength);
```

Description

This function takes as input image stored in ISO 19794-4 format and converts it into raw format. Please note that only version 2.0 of ISO 19794-4 format (ISO 19794-4:2011) is supported.

Parameters

`const unsigned char * isoFingerImage`

[in] Pointer to ISO 19794-4 image data

`unsigned int isoImageLength`

[in] Indicates length of input data

`int * width`

[out] The number of pixels indicating the width of the image as stored in ISO 19794-4 header.

`int * height`

[out] The number of pixels indicating the height of the image as stored in ISO 19794-4 header.

`unsigned char * fingerPosition`

[out] Indicates finger position as stored in ISO 19794-4 header.

`unsigned char * imageFormat`

[out] Indicates in which format is image data stored.

`unsigned int * dpiX`

[out] Horizontal resolution of the input image as stored in ISO 19794-4 header.

`unsigned int * dpiY`

[out] Vertical resolution of the input image as stored in ISO 19794-4 header.

`unsigned char * rawImage`

[out] Pointer where output raw data will be stored. The size of memory pointed by this parameter has to be at least equal to the value of the rawImageLength parameter.

`compressionRate`

[in] Indicates compression rate to be used in case of WSQ or JPEG2000 formats. For other formats, this parameter is ignored. Value of this parameter is interpreted as one over compression ratio (value 20 means compression rate of 1:20 etc.) Value of this parameter has to be greater or equal to 1.

`length`

[in/out] Pointer where output image (after background removal) will be stored. The size of memory pointed by this parameter has to be at least width*height bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

4.2.7 IEngine_ConvertRawToIso19794_4 Function

Converts raw image into ISO 19794-4:2011 (FIR version 2.0) format.

C++

```
IENGINE_API int IEngine_ConvertRawToIso19794_4(const unsigned char * rawImage, int width,
int height, unsigned char fingerPosition, unsigned char imageFormat, unsigned int dpiX,
unsigned int dpiY, unsigned char * outData, int rate, int * length);
```

Description

This function takes as input raw image, encodes this image in format specified and embeds it into ISO 19794-4 format. Please note that only version 2.0 of ISO 19794-4 format (ISO 19794-4:2011) is supported.

Parameters

`const unsigned char * rawImage`

[in] Pointer to the uncompressed raw image

`int width`

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

`int height`

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

`unsigned char fingerPosition`

[in] Indicates finger position. This information will be stored in the header of resulting ISO 19794-4 image data

`unsigned char imageFormat`

[in] Indicates desired image format used when creating ISO 19794-4 image data. Possible values: ISO_IMAGE_FORMAT_UNCOMPRESSED, ISO_IMAGE_FORMAT_WSQ, ISO_IMAGE_FORMAT_JPEG2000, ISO_IMAGE_FORMAT_PNG

`unsigned int dpiX`

[in] Indicates horizontal resolution of the input image. This information will be stored in the header of resulting ISO 19794-4 image data

`unsigned int dpiY`

[in] Indicates vertical resolution of the input image. This information will be stored in the header of resulting ISO 19794-4 image data

`unsigned char * outData`

[out] Pointer where output ISO 19794-4 image data will be stored. The size of memory pointed by this parameter has to be at least equal to the value of the length parameter.

`int rate`

[in] Specifies compression rate for JPEG2000 or bitrate for WSQ images. Ignored for other image formats. Value of this parameter for JPEG2000 format has to be greater or equal to 1 and is interpreted as one over compression ratio (value 20 means compression rate of 1:20 etc.). The meaningful range of parameter values for WSQ format is [1, 10]. To perform a correct conversion, the DPI resolution of the converting image must be 500.

`int * length`

[in/out] Pointer where output image (after background removal) will be stored. The size of memory pointed by this parameter has to be at least width*height bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

4.2.8 IEngine_ResizeImage Function

Reads raw 8-bit image from memory and creates new 8-bit 500dpi raw image.

C++

```
IENGINE_API int IEngine_ResizeImage(int width, int height, const BYTE * rawImage, int dpi,
int * outWidth, int * outHeight, BYTE * outImage);
```

Parameters

`int width`

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

`int height`

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

`const BYTE * rawImage`

[in] Pointer to the uncompressed raw image to resize

`int dpi`

[in] Input image dpi

`int * outWidth`

[out] The number of pixels indicating the width of the resized image

`int * outHeight`

[out] The number of pixels indicating the height of the resized image

`BYTE * outImage`

[out] Pointer to the uncompressed resized raw image

Returns

If outImage is NULL then the returned outWidth and outHeight parameters will indicate the size of memory needed to resize the image.

If outImage is not NULL then the image will be resized to outWidth = round(width * 500 / dpi) and outHeight = round(height * 500 / dpi).

Return Values

Return Values	Description
`IENGINE_E_NOERROR`	No error occurred.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_BADVALUE`	width, height or the resizeFactor is zero or negative.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

Related Topics

IEngine_ResizeImageInPlace (see page 14)

4.2.9 IEngine_ResizeImageInPlace Function

Reads raw 8-bit image from memory and resizes it to 500dpi in the same memory space.

C++

```
IENGINE_API int IEngine_ResizeImageInPlace(int width, int height, BYTE * rawImage, int dpi, int * outWidth, int * outHeight);
```

Parameters

`int width`

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

`int height`

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

`BYTE * rawImage`

[in/out] Pointer to the uncompressed raw image to resize

`int dpi`

[in] Input image dpi

`int * outWidth`

[out] The number of pixels indicating the width of the resized image

`int * outHeight`

[out] The number of pixels indicating the height of the resized image

Returns

If rawImage is NULL then the returned outWidth and outHeight parameters will indicate the size of memory needed to resize the image.

If rawImage is not NULL then rawImage will be resized to outWidth = round(width * 500 / dpi) and outHeight = round(height * 500 / dpi).

Return Values

Return Values	Description
`IENGINE_E_NOERROR`	No error occurred.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_BADVALUE`	width, height or the resizeFactor is zero or negative.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

Related Topics

IEngine_ResizeImage (see page 13)

4.2.10 ANSI_ConvertToISO Function

Converts ANSI/INCITS 378 compliant template to ISO/IEC 19794-2 compliant template

C++

```
IENGINE_API int ANSI_ConvertToISO(const BYTE * ansiTemplate, int * length, BYTE *
isoTemplate);
```

Description

This function converts an input ANSI/INCITS 378 compliant template to an output ISO/IEC 19794-2 compliant template. Templates with multiple finger views are supported by this function.

Parameters

`const BYTE * ansiTemplate`

[in] Reference ANSI/INCITS 378 template

`int * length`

[in/out] On input specifies the length of allocated data space (in bytes) pointed by isoTemplate pointer. On return, specifies the size of converted ISO template.

`BYTE * isoTemplate`

[out] Pointer to memory space where the resulting ISO/IEC 19794-2 compliant template will be written.

Returns

If isoTemplate parameter is NULL or if the size required to store the resulting template is more than the value specified by length parameter, total number of bytes required to store the resulting template is returned but no data is written to isoTemplate. If isoTemplate parameter is non-NULL and if the size required to store the resulting template is less or equal to the value specified by length parameter, both length parameter and isoTemplate are returned.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADTEMPLATE	The input templates is not valid ANSI/INCITS 378 template.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input or output parameter provided.

4.3 Template Extraction and Matching Functions

Functions

	Name	Description
	ANSI_CreateTemplate (see page 16)	Creates ANSI/INCITS 378 compliant template
	ANSI_CreateTemplate2 (see page 16)	Same as ANSI_CreateTemplate (see page 16) with possibility to specify input image resolution other than 500 dpi.
	ANSI_CreateTemplateEx (see page 17)	Creates ANSI/INCITS 378 compliant template, stores intermediate images
	ANSI_RemoveMinutiae (see page 18)	Removes minutiae points from an ANSI template
	ANSI_VerifyMatch (see page 19)	Compares two ANSI/INCITS 378 compliant templates
	ANSI_VerifyMatchEx (see page 19)	Compares specified finger views from ANSI/INCITS 378 compliant templates

4.3.1 ANSI_CreateTemplate Function

Creates ANSI/INCITS 378 compliant template

C++

```
IENGINE_API int ANSI_CreateTemplate(int width, int height, const BYTE * rawImage, BYTE * ansiTemplate);
```

Description

This function takes a raw image as input and generates the corresponding ANSI/INCITS 378 compliant fingerprint template. The memory for the template is allocated before the call (i.e., ANSI_CreateTemplate does not handle the memory allocation for the template parameter).

Parameters

`int width`

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

`int height`

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

`const BYTE * rawImage`

[in] Pointer to the uncompressed raw image for template creation

`BYTE * ansiTemplate`

[out] Pointer to memory space where the processed template will be written. Memory should be allocated before calling this function. The maximal size of generated template is 1568 bytes.

Returns

On success, ANSI/INCITS 378 compliant template is written to memory space pointed by ansiTemplate. Returned template contains default values in Record header and Finger View header. You may use ANSI_SetTemplateParameters in order to specify non-default values for Record header and Finger View header. To detect total length of created template, you may call function ANSI_GetTemplateParameter (see page 24) with PARAM_TEMPLATE_SIZE argument.

On failure, *NULL template* is returned.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADIMAGE	Image size is not in supported range.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable fingerprint. (Fail to enroll)
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Notes

NULL templates are defined as containing the Record header and Finger View header only, with zero minutiae information (i.e. Number of Minutiae shall be set to 0). Thus, it is a 32 byte template (26-byte Record Header + 4-byte Finger View header + 2 bytes for the Extended Data Block length which is 0x0000).

Related Topics

ANSI_SetTemplateParameter (see page 24), ANSI_GetTemplateParameter (see page 24)

4.3.2 ANSI_CreateTemplate2 Function

Same as ANSI_CreateTemplate (see page 16) with possibility to specify input image resolution other than 500 dpi.

C++

```
IENGINE_API int ANSI_CreateTemplate2(int width, int height, const BYTE * rawImage, int dpi,
BYTE * ansiTemplate);
```

Parameters

`int width`

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

`int height`

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

`const BYTE * rawImage`

[in] Pointer to the uncompressed raw image for template creation

`int dpi`

[in] Resolution (in dots per inch) of input image

`BYTE * ansiTemplate`

[out] Pointer to memory space where the processed template will be written. Memory should be allocated before calling this function. The maximal size of generated template is 1566 bytes.

Notes

Minutiae positions in output template will be resized to 500 dpi independent of the original resolution of the input image. If resolution of input image is different to 500 dpi, template dimensions (width, height) will not be the same as input image dimensions.

4.3.3 ANSI_CreateTemplateEx Function

Creates ANSI/INCITS 378 compliant template, stores intermediate images

C++

```
IENGINE_API int ANSI_CreateTemplateEx(int width, int height, const BYTE * rawImage, BYTE *
ansiTemplate, const char * skeletonImageFile, const char * binarizedImageFile, const char *
minutiaeImageFile);
```

Description

This function takes a raw image as input and generates the corresponding ANSI/INCITS 378 compliant fingerprint template. It optionally stores intermediate images produced during the extraction phase. The memory for the template is allocated before the call (i.e., ANSI_CreateTemplate (see page 16) does not handle the memory allocation for the template parameter).

Parameters

`int width`

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

`int height`

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

`const BYTE * rawImage`

[in] Pointer to the uncompressed raw image for template creation

`BYTE * ansiTemplate`

[out] Pointer to memory space where the processed template will be written. Memory should be allocated before calling this function. The maximal size of generated templates is 1568 bytes.

`const char * skeletonImageFile`

[in] Specifies the filename of bmp image where the fingerprint skeleton image will be saved. If this parameter is NULL, no skeleton image is saved.

`const char * binarizedImageFile`

[in] Specifies the filename of bmp image where the fingerprint binary image will be saved. If this parameter is NULL, no binary image is saved.

`const char * minutiaeImageFile`

[in] Specifies the filename of bmp image where the minutiae image will be saved (original fingerprint with all detected minutiae). If this parameter is NULL, no minutiae image is saved.

Returns

On success, ANSI/INCITS 378 compliant template is written to memory space pointed by ansiTemplate. Returned template

contains default values in Record header and Finger View header. You may use ANSI_SetTemplateParameters in order to specify non-default values for Record header and Finger View header.

On failure, *NULL template* is returned.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADIMAGE	Image size is not in supported range.
IENGINE_E_BLANKIMAGE	Image is blank or contains non-recognizable fingerprint. (Fail to enroll)
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input parameter provided.
IENGINE_E_NOTSUPPORTED_MSG	The function is not supported for this version of product.

Notes

NULL templates are defined as containing the Record header and Finger View header only, with zero minutiae information (i.e. Number of Minutiae shall be set to 0). Thus, it is a 32 byte template (26-byte Record Header + 4-byte Finger View header + 2 bytes for the Extended Data Block length which is 0x0000).

Related Topics

ANSI_SetTemplateParameter (see page 24)

4.3.4 ANSI_RemoveMinutiae Function

Removes minutiae points from an ANSI template

C++

```
IENGINE_API int ANSI_RemoveMinutiae(const BYTE * inTemplate, int maximumMinutiaeCount, int
* length, BYTE * outTemplate);
```

Description

This function limits maximal number of minutiae points contained in an ANSI/INCITS 378 compliant fingerprint template. If the input template contains more minutiae points than the provided limit, extra minutae point are removed from the template. The truncation is made by peeling off minutiae that are farthest from the point of gravity of the minutiae set.

Parameters

`const BYTE * inTemplate`

[in] Reference ANSI/INCITS 378 template

`int maximumMinutiaeCount`

[in] The maximal number of minutiae that will be stored in the output template. If current minutiae count is less or equal to this limit, the output template will be a copy of the input template.

`int * length`

[in/out] On input specifies the length of allocated data space (in bytes) pointed by outTemplate pointer. On return, specifies the size of truncated template

`BYTE * outTemplate`

[out] Pointer to memory space where the resulting truncated ANSI/INCITS 378 compliant template will be stored

Returns

If outTemplate parameter is NULL or if the size required to store the resulting template is more than the value specified by length parameter, total number of bytes required to store the resulting template is returned but no data is written to outTemplate. If outTemplate parameter is non-NULL and if the size required to store the resulting template is less or equal to the value specified by length parameter, both length parameter and outTemplate are returned.

Return Values

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

IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input parameter provided.

4.3.5 ANSI_VerifyMatch Function

Compares two ANSI/INCITS 378 compliant templates

C++

```
IENGINE_API int ANSI_VerifyMatch(const BYTE * probeTemplate, const BYTE * galleryTemplate,
int maxRotation, int * score);
```

Description

This function compares two ANSI/INCITS 378 compliant templates and outputs a match score. The score returned is an integer value ranging from 0 to 1000 which represents the similarity of original fingerprint images corresponding to compared templates. See topic Matching Scores for more details.

Parameters

`const BYTE * probeTemplate`
[in] ANSI/INCITS 378 template

`const BYTE * galleryTemplate`
[in] ANSI/INCITS 378 template

`int maxRotation`
[in] Maximal considered rotation between two fingerprint images. Valid range is between 0 and 180.

`int * score`
[out] On return, contains match score

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADTEMPLATE	At least one the input template is not valid ANSI/INCITS 378 template.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLTEMPLATE	At least one of the input templates is NULL (contains no finger view)
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Notes

Comparison in which either template is NULL template will result in match score equal to 0. If any input template contains multiple finger views only the first finger view (finger view with the lowest index number) is used for comparison. Use ANSI_VerifyMatchEx (see page 19) for comparing templates with multiple views.

This operation is not commutative, different order of templates will lead to a different matching score. For better understanding, the probe template (probeTemplate) determines how the templates will be matched. Changing the probe template for gallery template causes the templates will be matched slightly different way.

Related Topics

ANSI_VerifyMatchEx (see page 19)

4.3.6 ANSI_VerifyMatchEx Function

Compares specified finger views from ANSI/INCITS 378 compliant templates

C++

```
IENGINE_API int ANSI_VerifyMatchEx(const BYTE * probeTemplate, int probeView, const BYTE *
galleryTemplate, int galleryView, int maxRotation, int * score);
```

Description

This function compares given finger views from ANSI/INCITS 378 compliant templates and outputs a match score. The score returned is an integer value ranging from 0 to 1000 which represents the similarity of original fingerprint images corresponding to compared finger views. See topic Matching Scores for more details.

Parameters

`const BYTE * probeTemplate`
[in] ANSI/INCITS 378 template

`int probeView`
[in] Index number of the compared finger view from probe template. 0 is the first index number, 1 the second, etc.

`const BYTE * galleryTemplate`
[in] ANSI/INCITS 378 template

`int galleryView`
[in] Index number of the compared finger view from gallery template. 0 is the first index number, 1 the second, etc.

`int maxRotation`
[in] Maximal considered rotation between two fingerprint images. Valid range is between 0 and 180.

`int * score`
[out] On return, contains match score

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADPARAM	Specified finger view does not exist in the given template.
IENGINE_E_BADTEMPLATE	At least one the input template is not valid ANSI/INCITS 378 template.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLTEMPLATE	At least one of the input templates is NULL (contains no finger view)
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Notes

Comparison in which either template is NULL template will result in match score equal to 0.

This operation is not commutative, different order of templates will lead to a different matching score. For better understanding, the probe template (probeTemplate) determines how the templates will be matched. Changing the probe template for gallery template causes the templates will be matched slightly different way.

Related Topics

ANSI_VerifyMatch (see page 19), ANSI_GetTemplateParameter (see page 24)

4.4 Template Manipulation Functions

Functions

	Name	Description
	ANSI_DrawMinutiae (see page 21)	Returns bmp image with minutiae points marked over given fingerprint
	ANSI_MergeTemplates (see page 21)	Combines finger views from two ANSI/INCITS 378 templates into one common template
	ANSI_GetFingerView (see page 22)	Returns specified finger view from ANSI/INCITS 378 compliant template
	ANSI_LoadTemplate (see page 23)	Loads ANSI/INCITS 378 compliant template from file
	ANSI_SaveTemplate (see page 23)	Stores ANSI/INCITS 378 compliant template to file
	ANSI_SetTemplateParameter (see page 24)	Set specific template parameter
	ANSI_GetTemplateParameter (see page 24)	Get specific template parameters

4.4.1 ANSI_DrawMinutiae Function

Returns bmp image with minutiae points marked over given fingerprint

C++

```
IENGINE_API int ANSI_DrawMinutiae(const BYTE * ansiTemplate, int width, int height,
unsigned char * inputImage, unsigned char * outputBmpImage, int * outputImageLength);
```

Description

This function draws minutiae points associated with the given fingerprint template and returns the result as bmp image in memory. If inputImage is NULL, minutiae are drawn on a blank background, otherwise inputImage is used as background.

Parameters

`const BYTE * ansiTemplate`

[in] ANSI/INCITS 378 compliant fingerprint template

`int width`

[in] Width of inputImage, if inputImage is NULL, this value is ignored

`int height`

[in] Height of inputImage, if inputImage is NULL, this value is ignored

`unsigned char * inputImage`

[in] Raw image representing fingerprint from which fingerprint template was extracted

`unsigned char * outputBmpImage`

[out] A pointer to memory space where the resulting image will be stored. Resulting image will be stored in bmp format and its size will be specified in outputImageLength. If this value is NULL, image is not saved, only outputImageLength is set to required size.

`int * outputImageLength`

[in/out] On input, this value should indicate total allocated size of outputBmpImage. On output, this value will be equal to total memory size required to store resulting bmp image

Returns

If outputBmpImage is NULL or if outputImageLength is less than memory size of resulting bmp image, outputImageLength is set to total size (in bytes) of resulting bmp image Otherwise, outputImageLength is set to total size (in bytes) of resulting bmp image and bmp image is stored in memory space pointed by outputBmpImage. Resulting image is a color bmp image.

Return Values

Return Values	Description
`IEngine_ERRCODE_INVALID_DATA`	indicates that the input template has not correct format or is damaged

4.4.2 ANSI_MergeTemplates Function

Combines finger views from two ANSI/INCITS 378 templates into one common template

C++

```
IENGINE_API int ANSI_MergeTemplates(const BYTE * referenceTemplate, const BYTE *
addedTemplate, int * length, BYTE * outTemplate);
```

Description

This function reads finger views from two ANSI/INCITS 378 templates and merges them into one resulting template. Each input template may contain zero, one or multiple finger views. Record header of the resulting template is copied from the record header of the reference template.

Parameters

`const BYTE * referenceTemplate`

[in] Reference ANSI/INCITS 378 template

`const BYTE * addedTemplate`

[in] ANSI/INCITS 378 template. Finger views from this template will be combined with finger views from the reference template.

`int * length`

[in/out] On input specifies the length of allocated data space (in bytes) pointed by outTemplate pointer. On return, specifies the size of merged template containing all finger views.

`BYTE * outTemplate`

[out] Pointer to memory space where the resulting ANSI/INCITS 378 compliant template containing all finger views from input templates will be written.

Returns

If outTemplate parameter is NULL or if the size required to store the resulting template is more than the value specified by length parameter, total number of bytes required to store the resulting template is returned but no data is written to outTemplate. If outTemplate parameter is non-NULL and if the size required to store the resulting template is less or equal to the value specified by length parameter, both length parameter and outTemplate are returned.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADTEMPLATE	At least one of the input templates is not valid ANSI/INCITS 378 template.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input or output parameter provided.

Related Topics

ANSI_GetFingerView (see page 22)

4.4.3 ANSI_GetFingerView Function

Returns specified finger view from ANSI/INCITS 378 compliant template

C++

```
IENGINE_API int ANSI_GetFingerView(const BYTE * ansiTemplate, int fingerView, BYTE *
outTemplate);
```

Description

This function reads specified finger view from given ANSI/INCITS 378 compliant template and returns new ANSI/INCITS 378 compliant template containing only single finger view. Record header of the new template is a copy of the record header of the input template.

Parameters

`const BYTE * ansiTemplate`

[in] ANSI/INCITS 378 template

`int fingerView`

[in] Index number of the finger view that will be returned in the newly created template. 0 is the first index number, 1 the second, etc.

`BYTE * outTemplate`

[out] Pointer to memory space where the resulting ANSI/INCITS 378 template containing the specified finger view will be written. The maximal size of such template is 1568 bytes.

Returns

On success, ANSI/INCITS 378 compliant template containing single finger view is written to memory space pointed by outTemplate. Returned template contains same values in Record header as the input template.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADPARAM	Specified finger view does not exist in the given template.
IENGINE_E_BADTEMPLATE	The input template is not valid ANSI/INCITS 378 template.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLTEMPLATE	The input template is NULL (contains no finger view)

IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input or output parameter provided.

Related Topics

ANSI_MergeTemplates (see page 21)

4.4.4 ANSI_LoadTemplate Function

Loads ANSI/INCITS 378 compliant template from file

C++

```
IENGINE_API int ANSI_LoadTemplate(const char * filename, BYTE * ansiTemplate);
```

Description

This function loads an ANSI/INCITS 378 compliant template from file

Parameters

`const char * filename`

[in] Name of the file where the template will be saved

`BYTE * ansiTemplate`

[out] Pointer to memory space where the ANSI/INCITS 378 compatible template will be loaded. Memory should be allocated before calling this function. The maximal size of ANSI/INCITS 378 template with one finger view is 1568 bytes. Consider number of finger views in the template (merged template). The maximal size of one finger view is 1542 bytes.

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.
IENGINE_E_BADTEMPLATE	Input template is not valid ANSI/INCITS 378 template.
IENGINE_E_FILE	Error occurred while opening/reading file.

Related Topics

ANSI_GetTemplateParameter (see page 24)

4.4.5 ANSI_SaveTemplate Function

Stores ANSI/INCITS 378 compliant template to file

C++

```
IENGINE_API int ANSI_SaveTemplate(const char * filename, const BYTE * ansiTemplate);
```

Description

This function stores the input ANSI/INCITS 378 compliant template to a file

Parameters

`const char * filename`

[in] Name of the file where the input template will be saved

`const BYTE * ansiTemplate`

[in] ANSI/INCITS 378 template

Return Values

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

IENGINE_E_NULLPARAM	NULL input parameter provided.
IENGINE_E_BADTEMPLATE	Input template is not valid ANSI/INCITS 378 template.
IENGINE_E_FILE	Error occurred while opening/reading file.

Related Topics

ANSI_SetTemplateParameter (see page 24)

4.4.6 ANSI_SetTemplateParameter Function

Set specific template parameter

C++

```
IENGINE_API int ANSI_SetTemplateParameter(BYTE * ansiTemplate, IENGINE_TEMPLATE_PARAMETER parameter, int value);
```

Description

This function modifies the information concerning specific template parameters stored in record header or in finger view header of an ANSI/INCITS 378 template. If specified template contains multiple finger views, this function modifies the first finger view (finger view with the lowest index number).

Parameters

`BYTE * ansiTemplate`

[in/out] On input, this parameter should contain a valid ANSI/INCITS 378 template. On return, the original template is modified - .

`IENGINE_TEMPLATE_PARAMETER parameter`

[in] Contains the code of the template parameter to be set.

`int value`

[in] Contains the value for the specified parameter

Return Values

Return Values	Description
IENGINE_E_NOERROR	No error occurred.
IENGINE_E_BADPARAM	Invalid parameter type provided.
IENGINE_E_BADTEMPLATE	Input template is not valid ANSI/INCITS 378 template.
IENGINE_E_NULLPARAM	NULL input parameter provided.
IENGINE_E_BADVALUE	Invalid value provided.
IENGINE_E_READONLY	Specified parameter cannot be modified.
IENGINE_E_NOTDEFINED	Specified parameter is not defined (NULL template?)

Related Topics

ANSI_GetTemplateParameter (see page 24)

4.4.7 ANSI_GetTemplateParameter Function

Get specific template parameters

C++

```
IENGINE_API int ANSI_GetTemplateParameter(const BYTE * ansiTemplate, IENGINE_TEMPLATE_PARAMETER parameter, int * value);
```

Description

This function retrieves the value of a specific template parameter stored in record header or in finger view header of the input ANSI/INCITS 378 template. If specified template contains multiple finger views, this function retrieves information related to the first finger view (finger view with the lowest index number).

Parameters

`const BYTE * ansiTemplate`
[in] ANSI/INCITS 378 template

`IENGINE_TEMPLATE_PARAMETER parameter`
[in] Contains the code of the template parameter

`int * value`
[out] On return, contains the 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_BADTEMPLATE`	Input template is not valid ANSI/INCITS 378 template.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.
`IENGINE_E_NOTDEFINED`	Specified parameter is not defined (NULL template?)

Related Topics

ANSI_SetTemplateParameter (see page 24)

5 Types, Structures, Enumerations

Enumerations

Name	Description
IENGINE_HWID_METHOD (see page 26)	• Summary Enumeration defining codes for different hardware ID methods.
IENGINE_FINGER_POSITION (see page 27)	Enumeration defining codes for different finger positions
IENGINE_IMAGE_FORMAT (see page 27)	Structure representing a supported image formats.

5.1 IENGINE_HWID_METHOD Enumeration

C++

```
typedef enum {
  IENGINE_HWID_METHOD_AUTO = 0,
  IENGINE_HWID_METHOD_DISKID = 1,
  IENGINE_HWID_METHOD_MAC = 2,
  IENGINE_HWID_METHOD_SERIALNO = 3,
  IENGINE_HWID_METHOD_IMEI = 4,
  IENGINE_HWID_METHOD_SMBIOS = 5,
  IENGINE_HWID_METHOD_AMAZON = 6,
  IENGINE_HWID_METHOD_APPID = 7,
  IENGINE_HWID_METHOD_PHY = 8,
  IENGINE_HWID_METHOD_MAC_ADDR = 9
} IENGINE_HWID_METHOD;
```

Description

This enumeration is used in IEngine_GetHwidByMethod (see page 6)function

Members

Members	Description
IENGINE_HWID_METHOD_AUTO = 0	Automatically selects the hardware ID method based on platform defaults On Linux platforms, IENGINE_HWID_METHOD_MAC is being used On Windows platforms, IENGINE_HWID_METHOD_DISKID is being used On Android platforms, IENGINE_HWID_METHOD_APPID is being used
IENGINE_HWID_METHOD_DISKID = 1	Use disk based hardware id This method is available only on Windows platforms
IENGINE_HWID_METHOD_MAC = 2	Use network MAC address based hardware ID This method is available only on Linux platforms
IENGINE_HWID_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_HWID_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_HWID_METHOD_SMBIOS = 5	Use SMBIOS UUID based hardware ID • Not supported
IENGINE_HWID_METHOD_AMAZON = 6	Use application ID based hardware ID • Available on Android platforms
IENGINE_HWID_METHOD_APPID = 7	Use application ID based hardware ID • Available on Android platforms
IENGINE_HWID_METHOD_PHY = 8	Use only network mac address as hardware ID • Available on ARM/Android ("IM00" form)

`IENGINE_HWID_METHOD_MAC_ADDR = 9`	Use only network mac address as hardware ID • Available on ARM/Android ("IM00" form)

Remarks

- Summary

Enumeration defining codes for different hardware ID methods.

5.2 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
} IENGINE_FINGER_POSITION;
```

5.3 IENGINE_IMAGE_FORMAT Enumeration

C++

```
typedef enum {
  IENGINE_FORMAT_BMP = 0,
  IENGINE_FORMAT_PNG = 1,
  IENGINE_FORMAT_WSQ = 5,
  IENGINE_FORMAT_JPEG2K = 6
} IENGINE_IMAGE_FORMAT;
```

Remarks

Structure representing a supported image formats.

6 Constants

7 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_GetErrorMessage API call.

Error	Error Code	Description
IENGINE_E_NOERROR	0	No error.
IENGINE_E_BADPARAM	1101	Invalid parameter type provided.
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/reading file.
IENGINE_E_MEMORY	1120	Memory allocation failed.
IENGINE_E_NULLPARAM	1121	NULL input parameter provided.
IENGINE_E_OTHER	1122	Other unspecified error.
IENGINE_E_NOTSUPPORTED	1123	The function is not supported for this version of product.
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_READONLY	1136	Value cannot be modified
IENGINE_E_NOTDEFINED	1137	Value is not defined
IENGINE_E_NULLTEMPLATE	1138	Provided template is NULL (contains no finger view)
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 1800x1800.
IENGINE_E_BADDPI	1148	Image does not have required DPI.

8 Symbol Reference

8.1 Functions

The following table lists functions in this documentation.

8.2 Structs, Records, Enums

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

9 Notes

9.1 Fingerprint Quality

ANSI&ISO SDK provides API for fingerprint image quality acquisition. Image quality number is calculated in accordance with the general guidelines contained in Section 7.2.5 of ANSI/INCITS 381 standard (Section 8.3.7.3 Quality score of ISO/IEC 19794-4 international standard).

Fingerprint image quality is linked to the total surface of ridges found in the fingerprint ("usable" surface area which is not classified as background noise). Only ridges that are well defined and can be extracted safely are counted, if a ridge belongs to the background (noisy) zone, it is not considered. Total ridge surface is normalized by the total sensor/fingerprint image surface, fingerprint quality is therefore a number between 0 and 100.

Fingerprint image quality can be obtained using:

- IEngine_GetImageQuality Function (see page 7)
- PARAM_FINGER_QUALITY Data Member of IENGINE_TEMPLATE_PARAMETER Enumeration

9.2 Fingerprint Template Format

ANSI/ISO SDK supports various fingerprint template formats, For more information please refer to IENGINE_TEMPLATE_FORMAT Enumeration. Templates can be converted via following functions:

- IEngine_ConvertTemplate Function
- ANSI_ConvertToISO Function (see page 15) and ISO_ConvertToANSI Function for ANSI INCITS 378 <-> ISO/IEC 19794-2:2005 conversions
- ISO_ConvertToISOCardCC Function and ISO_CARD_CC_ConvertToISO Function for ISO/IEC 19794-2:2005 <-> ISO CC conversions

For matching only ANSI INCITS 378 or ISO/IEC 19794-2:2005 templates can be used.

9.3 Threshold

The table below gives an indication which similarity threshold corresponds to which

FAR (False Accept Rate). Please note that the dependence between threshold

and FAR may be dependent on factors such as scanner type, quality of the data, etc.

The below numbers were obtained on a sample database with 200,000 non-corresponding

fingerprint pairs. Fingerprints were acquired with an optical 500 DPI scanner.

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..)

Threshold at FAR = 1% (1:100)	Threshold at FAR = 0.1% (1:1,000)	Threshold at FAR = 0.01% (1:10,000)	Threshold at FAR = 0.001% (1:100,000)
20	30	40	50

10 Open source SW credits

Some of the components included in our SDK are licensed under free or open source licenses. We wish to thank the contributors to those projects.

10.1 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.
```

10.2 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.
```

10.3 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
```

10.4 zlib

http://www.zlib.net/zlib_license.html

```
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
```

10.5 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.
```

10.6 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.
```

10.7 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."
```

10.8 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.
```

10.9 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.
```

Index

T

Z