Innovatrics ANSI/ISO Generator & Matcher

Table of Contents

1 Overview

The Innovatrics ANSI/ISO Generator & Matcher is a standard windows dll library designed to

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

Innovatrics ANSI/ISO Generator takes as input raw fingerprint images and transforms them to ANSI/ISO compliant templates.

Innovatrics ANSI/IISO Matcher takes as input two ANSI/ISO compliant templates and returns corresponding similarity score.

The API of the library is based on MINEX API specifications designed by NIST.

All functions from this library are thread-safe.

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.

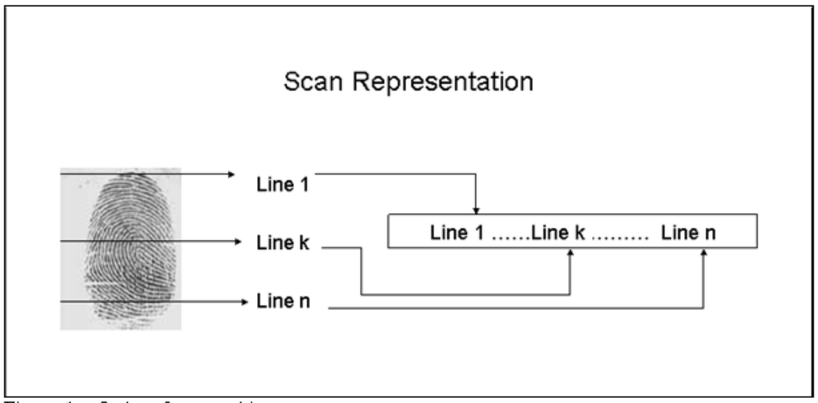

Figure 1 – Order of scanned images

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 resolution of images supplied to the Generator should be 500 DPI. The dimensions of the fingerprint images may vary from 150 to 1000 pixels in width, and 166 to 1000 pixels in height.

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

3 Matching Scores

Similarity scores returned by matching functions (ANSI_VerifyMatch (see page 14), ANSI_VerifyMatchEx (see page 15), ISO_VerifyMatch (see page 17), ISO_VerifyMatchEx (see page 18)) are integer values ranging from 0 to 100000. 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 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. The table below shows different thresholds and observed FAR and FRR rates. This table was generated using a sample database of 500 DPI images from optical sensors.

Threshold	False Accept Rate (FAR)	False Reject Rate (FRR)
9500	1% (1:100)	0.09%
11600	0.1% (1:1.000)	0.25%
13700	0.01% (1:10.000)	0.55%
15650	0.001% (1:100.000)	0.9%
17750	0.0001% (1:10^6)	1.45%

4 Library Functions

4.1 Init, Terminate and other General Functions

Functions

	Name	Description
	IEngine_Init (see page 4)	Initializes the library
	IEngine_Terminate (see page 4)	Terminates the use of the library
	IEngine_GetVersion (see page 5)	Returns the library version
	IEngine_GetErrorMessage (see page 5)	Returns human understandable error message
	IEngine_GetImageQuality (see page 5)	Returns quality of a fingerprint image
	IEngine_SetLicenseContent (see page 6)	Activates the library with a license key

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_OK`	No error occurred.
`IENGINE_E_BADLICENSE`	Provided license is not valid, or no license was found.

Related Topics

IEngine_Terminate (see page 4), IEngine_GetVersion (see page 5)

4.1.2 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_OK`	No error occurred.

Related Topics

IEngine_Init (see page 4)

4.1.3 IEngine_GetVersion Function

Returns the library version

C++

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

Description

This function returns the version number of the library

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

Version

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

4.1.4 IEngine_GetErrorMessage Function

Returns human understandable error message

C++

```
IENGINE_API char * IEngine_GetErrorMessage(int errcode);
```

Description

This function returns error message string corresponding to the specified error code.

Parameters

`int errcode`

[in] Error code to be translated into error message

Related Topics

Error Codes (see page 37)

4.1.5 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 2.1.42 of ANSI/INCITS 358 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)

4.1.6 IEngine_SetLicenseContent Function

Activates the library with a license key

C++

```
IENGINE_API int IEngine_SetLicenseContent(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.

Parameters

`unsigned char * licenseContent`

[in] License information as provided by Innovatrics

`int length`

[in] Total length of license data

Return Values

Return Values	Description
IENGINE_OK	No error occured.
IENGINE_E_BADLICENSE	Invalid license data

4.2 Conversion Functions

Functions

	Name	Description
◆	ANSI_ConvertToISO (see page 6)	Converts ANSI/INCITS 378 compliant template to ISO/IEC 19794-2 compliant template
◆	ISO_CARD_CC_ConvertToISO (see page 7)	Converts an ISO Compact Card template into regular ISO template
◆	ISO_ConvertToANSI (see page 8)	Converts ISO/IEC 19794-2 compliant template to ANSI/INCITS 378 compliant template
◆	ISO_ConvertToISOCardCC (see page 8)	Converts a regular ISO template to ISO Compact Card template format (ISO Compact Size Finger Minutiae Format)
◆	IEngine_ConvertBMP (see page 9)	Reads bmp image from memory and converts it to raw 8-bit format
◆	IEngine_ConvertTemplate (see page 10)	Converts between different template formats.
◆	IEngine_LoadBMP (see page 11)	Loads bmp image from file and converts it to raw 8-bit format

4.2.1 ANSI_ConvertToISO Function

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

C++

```
IENGINE_API int ANSI_ConvertToISO(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

`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_OK	No error occured.
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.

4.2.2 ISO_CARD_CC_ConvertToISO Function

Converts an ISO Compact Card template into regular ISO template

C++

```
IENGINE_API int ISO_CARD_CC_ConvertToISO(BYTE * isoCCTemplate, int * length, BYTE *
isoTemplate);
```

Description

This function takes as input a Finger Minutiae Compact Card Format template (ISO_CARD_CC template) and converts it into ISO/IEC 19794-2 compliant fingerprint template. An ISO_CARD_CC template is defined in ISO/IEC 19794-2 standard, under paragraph 8. In ISO_CARD_CC template, each minutiae point is encoded in 3 bytes (instead of 6 bytes used in regular ISO template).

Parameters

`BYTE * isoCCTemplate`

[in] ISO Compact Card 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 the resulting ISO template

`BYTE * isoTemplate`

[out] Pointer to memory space where the resulting ISO template will be stored

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_OK`	No error occured.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_MEMORY`	Memory allocation failed.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

Related Topics

ISO_ConvertToISOCardCC (see page 8)

4.2.3 ISO_ConvertToANSI Function

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

C++

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

Description

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

Parameters

`BYTE * isoTemplate`

[in] Reference ISO/IEC 19794-2 template

`int * length`

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

`BYTE * ansiTemplate`

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

Returns

If ansiTemplate 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 ansiTemplate. If ansiTemplate 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 ansiTemplate are returned.

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`IENGINE_E_BADTEMPLATE`	The input templates is not valid ISO/IEC 19794-2 template.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_MEMORY`	Memory allocation failed.
`IENGINE_E_NULLPARAM`	NULL input or output parameter provided.

4.2.4 ISO_ConvertToISOCardCC Function

Converts a regular ISO template to ISO Compact Card template format (ISO Compact Size Finger Minutiae Format)

C++

```
IENGINE_API int ISO_ConvertToISOCardCC(BYTE * isoTemplate, int maximumMinutiaeCount,
IENGINE_SORT_ORDER minutiaeOrder, IENGINE_SORT_ORDER minutiaeSecondaryOrder, int * length,
BYTE * isoCCTemplate);
```

Description

This function takes as input an ISO/IEC 19794-2 compliant fingerprint template and converts it into Finger Minutiae Compact Card Format template (ISO_CARD_CC template). An ISO_CARD_CC template is defined in ISO/IEC 19794-2 standard, under paragraph 8. In ISO_CARD_CC template, each minutiae point is encoded in 3 bytes (instead of 6 bytes used in regular ISO template). This function can also truncate minutiae points, if the number of minutiae points in the template is too big. You may also specify desired minutiae order in which minutiae points should be stored in the template.

Parameters

`BYTE * isoTemplate`

[in] Reference ISO/IEC 19794-2 template

`int maximumMinutiaeCount`

[in] The maximal number of minutiae that will be stored in the output template. See ISO_RemoveMinutiae (see page 29) for details about truncation algorithm.

`IENGINE_SORT_ORDER minutiaeOrder`

[in] Defines the primary ordering criteria for minutiae

`IENGINE_SORT_ORDER minutiaeSecondaryOrder`

[in] Defines the secondary ordering criteria for minutiae (used when primary ordering criteria gives equality)

`int * length`

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

`BYTE * isoCCTemplate`

[out] Pointer to memory space where the resulting ISO Compact Card template will be stored

Returns

If isoCCTemplate 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 isoCCTemplate. If isoCCTemplate 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 isoCCTemplate are returned.

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_MEMORY`	Memory allocation failed.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

Related Topics

ISO_RemoveMinutiae (see page 29), ISO_CARD_CC_ConvertToISO (see page 7)

4.2.5 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_OK	No error occured.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_FILE	Error occured while opening/reading file.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Related Topics

IEngine_LoadBMP (see page 11)

4.2.6 IEngine_ConvertTemplate Function

Converts between different template formats.

C++

```
IENGINE_API int IEngine_ConvertTemplate(IENGINE_TEMPLATE_FORMAT inputTemplateType, BYTE *
inputTemplate, IENGINE_TEMPLATE_FORMAT outputTemplateType, int * length, BYTE *
outputTemplate);
```

Description

This function converts an input template to an equivalent template in a format specified by user. Templates with multiple finger views are supported by this function. When converting to ILO_SID_TEMPLATE format, all "fixed" values are set according to the specification (see http://www.ilo.org/public/english/dialogue/sector/papers/maritime/sid0002.pdf Annex B for more details). Similarly, the resulting template will always have two fingers. If the input template contains only one (or none) fingerview, the resulting ILO SID template will have one (or two) "unenrolled" fingers with finger quality set to 0x65 (enrollment failed due to a physical disability).

Parameters

`IENGINE_TEMPLATE_FORMAT inputTemplateType`

[in] Specifies the format of the input template

`BYTE * inputTemplate`

[in] Reference template

`IENGINE_TEMPLATE_FORMAT outputTemplateType`

[in] Specifies the format of the output template

`int * length`

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

`BYTE * outputTemplate`

[out] Pointer to memory space where the resulting template will be written.

Returns

If outputTemplate 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 outputTemplate. If outputTemplate 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 outputTemplate are returned.

Return Values

Return Values	Description
IENGINE_OK	No error occured.
IENGINE_E_BADTEMPLATE	The input templates is not valid or has a different format as specified.
IENGINE_E_BADPARAM	Invalid input parameter provided (invalid template format?)
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_MEMORY	Memory allocation failed.
IENGINE_E_NULLPARAM	NULL input or output parameter provided.

4.2.7 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_OK	No error occured.
IENGINE_E_BADFORMAT	Unsupported image format.
IENGINE_E_FILE	Error occured while opening/reading file.
IENGINE_E_INIT	Library was not initialized.
IENGINE_E_NULLPARAM	NULL input parameter provided.

Related Topics

IEngine_ConvertBMP (see page 9)

4.3 Template Extraction and Matching Functions

Functions

	Name	Description
	ANSI_CreateTemplate (see page 12)	Creates ANSI/INCITS 378 compliant template
	ANSI_CreateTemplateEx (see page 13)	Creates ANSI/INCITS 378 compliant template, stores intermediate images
	ANSI_VerifyMatch (see page 14)	Compares two ANSI/INCITS 378 compliant templates
	ANSI_VerifyMatchEx (see page 15)	Compares specified finger views from ANSI/INCITS 378 compliant templates
	ISO_CreateTemplate (see page 15)	Creates ISO/IEC 19794-2 compliant template
	ISO_CreateTemplateEx (see page 16)	Creates ISO/IEC 19794-2 compliant template, stores intermediate images
	ISO_VerifyMatch (see page 17)	Compares two ISO/IEC 19794-2 compliant templates
	ISO_VerifyMatchEx (see page 18)	Compares specified finger views from ISO/IEC 19794-2 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 21) with PARAM_TEMPLATE_SIZE argument.

On failure, *NULL template* is returned.

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`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 21)

4.3.2 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 12) 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_OK`	No error occured.
`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)

4.3.3 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 100000 which represents the similarity of original fingerprint images corresponding to compared templates. See topic Matching Scores (see page 3) 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_OK`	No error occured.
`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 15) for comparing templates with multiple views.

Related Topics

ANSI_VerifyMatchEx (see page 15)

4.3.4 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 100000 which represents the similarity of original fingerprint images corresponding to compared finger views. See topic Matching Scores (see page 3) 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_OK	No error occured.
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.

Related Topics

ANSI_VerifyMatch (see page 14), ANSI_GetTemplateParameter (see page 21)

4.3.5 ISO_CreateTemplate Function

Creates ISO/IEC 19794-2 compliant template

C++

```
IENGINE_API int ISO_CreateTemplate(int width, int height, const BYTE * rawImage, BYTE *
isoTemplate);
```

Description

This function takes a raw image as input and generates the corresponding ISO/IEC 19794-2 compliant fingerprint template. The memory for the template is allocated before the call (i.e., ISO_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 * isoTemplate`

[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.

Returns

On success, ISO/IEC 19794-2 compliant template is written to memory space pointed by isoTemplate. Returned template contains default values in Record header and Finger View header. You may use ISO_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 ISO_GetTemplateParameter (see page 27) with PARAM_TEMPLATE_SIZE argument.

On failure, *NULL template* is returned.

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`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 30 byte template (24-byte Record Header + 4-byte Finger View header + 2 bytes for the Extended Data Block length which is 0x0000).

Related Topics

ISO_SetTemplateParameter (see page 30), ISO_GetTemplateParameter (see page 27)

4.3.6 ISO_CreateTemplateEx Function

Creates ISO/IEC 19794-2 compliant template, stores intermediate images

C++

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

Description

This function takes a raw image as input and generates the corresponding ISO/IEC 19794-2 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., ISO_CreateTemplate (see page 15) 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 * isoTemplate`

[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 1566 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, ISO/IEC 19794-2 compliant template is written to memory space pointed by isoTemplate. Returned template contains default values in Record header and Finger View header. You may use ISO_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_OK`	No error occured.
`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 30 byte template (24-byte Record Header + 4-byte Finger View header + 2 bytes for the Extended Data Block length which is 0x0000).

Related Topics

ISO_SetTemplateParameter (see page 30)

4.3.7 ISO_VerifyMatch Function

Compares two ISO/IEC 19794-2 compliant templates

C++

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

Description

This function compares two ISO/IEC 19794-2 compliant templates and outputs a match score. The score returned is an integer value ranging from 0 to 100000 which represents the similarity of original fingerprint images corresponding to compared templates. See topic Matching Scores (see page 3) for more details.

Parameters

`const BYTE * probeTemplate`

[in] ISO/IEC 19794-2 template

`const BYTE * galleryTemplate`

[in] ISO/IEC 19794-2 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_OK`	No error occured.
`IENGINE_E_BADTEMPLATE`	At least one the input template is not valid ISO/IEC 19794-2 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 ISO_VerifyMatchEx (see page 18) for comparing templates with multiple views.

Related Topics

ISO_VerifyMatchEx (see page 18)

4.3.8 ISO_VerifyMatchEx Function

Compares specified finger views from ISO/IEC 19794-2 compliant templates

C++

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

Description

This function compares given finger views from ISO/IEC 19794-2 compliant templates and outputs a match score. The score returned is an integer value ranging from 0 to 100000 which represents the similarity of original fingerprint images corresponding to compared finger views. See topic Matching Scores (see page 3) for more details.

Parameters

`const BYTE * probeTemplate`

[in] ISO/IEC 19794-2 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] ISO/IEC 19794-2 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

4

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`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 ISO/IEC 19794-2 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.

Related Topics

ISO_VerifyMatch (see page 17), ISO_GetTemplateParameter (see page 27)

4.4 Template Manipulation Functions

Functions

	Name	Description
	ANSI_DrawMinutiae (see page 19)	Returns bmp image with minutiae points marked over given fingerprint
	ANSI_GetFingerView (see page 20)	Returns specified finger view from ANSI/INCITS 378 compliant template
	ANSI_GetMinutiae (see page 21)	Returns minutiae stored in ANSI/INCITS 378 compliant template
	ANSI_GetTemplateParameter (see page 21)	Get specific template parameters
	ANSI_MergeTemplates (see page 22)	Combines finger views from two ANSI/INCITS 378 templates into one common template
	ANSI_LoadTemplate (see page 22)	Loads ANSI/INCITS 378 compliant template from file
	ANSI_RemoveMinutiae (see page 23)	Removes minutiae points from an ANSI template
	ANSI_SaveTemplate (see page 24)	Stores ANSI/INCITS 378 compliant template to file
	ANSI_SetTemplateParameter (see page 24)	Set specific template parameter
	ISO_DrawMinutiae (see page 25)	Returns bmp image with minutiae points marked over given fingerprint
	ISO_GetFingerView (see page 25)	Returns specified finger view from ISO/IEC 19794-2 compliant template
	ISO_GetMinutiae (see page 26)	Returns minutiae stored in ISO/IEC 19794-2 compliant template
	ISO_GetTemplateParameter (see page 27)	Get specific template parameters
	ISO_LoadTemplate (see page 27)	Loads ISO/IEC 19794-2 compliant template from file
	ISO_MergeTemplates (see page 28)	Combines finger views from two ISO/IEC 19794-2 templates into one common template
	ISO_RemoveMinutiae (see page 29)	Removes minutiae points from an ISO template
	ISO_SaveTemplate (see page 29)	Stores ISO/IEC 19794-2 compliant template to file
	ISO_SetTemplateParameter (see page 30)	Set specific template parameter

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_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_OK`	No error occured.
`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 22)

4.4.3 ANSI_GetMinutiae Function

Returns minutiae stored in ANSI/INCITS 378 compliant template

C++

```
IENGINE_API int ANSI_GetMinutiae(const BYTE * ansiTemplate, IENGINE_MINUTIAE minutiae[256],
int * minutiaeCount);
```

Description

This function returns minutiae angles and minutiae positions stored in ANSI/INCITS 378 compliant template

Parameters

`const BYTE * ansiTemplate`

[in] ANSI/INCITS 378 template

`IENGINE_MINUTIAE minutiae[256]`

[out] Initialized minutiae array with 256 elements (maximal minutiae count in ANSI/INCITS 378 template is 256). If null, minutiae points will not be returned.

`int * minutiaeCount`

[out] Minutiae count contained within the input template

Returns

On success, minutiae and minutiaeCount are returned. If either of those parameter is NULL, the parameter is not returned.

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`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_NULLTEMPLATE`	Input template is NULL (contains no finger view)

4.4.4 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_OK`	No error occured.

`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)

4.4.5 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_OK`	No error occured.
`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 20)

4.4.6 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 template is 1568 bytes.

Return Values

Return Values	Description
IENGINE_OK	No error occured.
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 occured while opening/reading file.

Related Topics

ANSI_GetTemplateParameter (see page 21)

4.4.7 ANSI_RemoveMinutiae Function

Removes minutiae points from an ANSI template

C++

```
IENGINE_API int ANSI_RemoveMinutiae(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

`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_OK	No error occured.

`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_MEMORY`	Memory allocation failed.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

4.4.8 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_OK`	No error occured.
`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 occured while opening/reading file.

Related Topics

ANSI_SetTemplateParameter (see page 24)

4.4.9 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_OK`	No error occured.

4

`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 21)

4.4.10 ISO_DrawMinutiae Function

Returns bmp image with minutiae points marked over given fingerprint

C++

```
IENGINE_API int ISO_DrawMinutiae(const BYTE * isoTemplate, 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 * isoTemplate`

[in] ISO/IEC 19794-2 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

4.4.11 ISO_GetFingerView Function

Returns specified finger view from ISO/IEC 19794-2 compliant template

C++

```
IENGINE_API int ISO_GetFingerView(const BYTE * isoTemplate, int fingerView, BYTE *
outTemplate);
```

Description

This function reads specified finger view from given ISO/IEC 19794-2 compliant template and returns new ISO/IEC 19794-2 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 * isoTemplate`

[in] ISO/IEC 19794-2 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 ISO/IEC 19794-2 template containing the specified finger view will be written. The maximal size of such template is 1566 bytes.

Returns

On success, ISO/IEC 19794-2 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_OK`	No error occured.
`IENGINE_E_BADPARAM`	Specified finger view does not exist in the given template.
`IENGINE_E_BADTEMPLATE`	The input template is not valid ISO/IEC 19794-2 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

ISO_MergeTemplates (see page 28)

4.4.12 ISO_GetMinutiae Function

Returns minutiae stored in ISO/IEC 19794-2 compliant template

C++

```
IENGINE_API int ISO_GetMinutiae(const BYTE * isoTemplate, IENGINE_MINUTIAE minutiae[256],
int * minutiaeCount);
```

Description

This function returns minutiae angles and minutiae positions stored in ISO/IEC 19794-2 compliant template

Parameters

`const BYTE * isoTemplate`

[in] ISO/IEC 19794-2 template

`IENGINE_MINUTIAE minutiae[256]`

[out] Initialized minutiae array with 256 elements (maximal minutiae count in ISO/IEC 19794-2 template is 256). If null, minutiae points will not be returned.

`int * minutiaeCount`

[out] Minutiae count contained within the input template

Returns

On success, minutiae and minutiaeCount are returned. If either of those parameter is NULL, the parameter is not returned.

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.

`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.
`IENGINE_E_BADTEMPLATE`	Input template is not valid ISO/IEC 19794-2 template.
`IENGINE_E_NULLTEMPLATE`	Input template is NULL (contains no finger view)

4.4.13 ISO_GetTemplateParameter Function

Get specific template parameters

C++

```
IENGINE_API int ISO_GetTemplateParameter(const BYTE * isoTemplate,
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 ISO/IEC 19794-2 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 * isoTemplate`

[in] ISO/IEC 19794-2 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_OK`	No error occured.
`IENGINE_E_BADPARAM`	Invalid parameter type provided.
`IENGINE_E_BADTEMPLATE`	Input template is not valid ISO/IEC 19794-2 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

ISO_SetTemplateParameter (see page 30)

4.4.14 ISO_LoadTemplate Function

Loads ISO/IEC 19794-2 compliant template from file

C++

```
IENGINE_API int ISO_LoadTemplate(const char * filename, BYTE * isoTemplate);
```

Description

This function loads an ISO/IEC 19794-2 compliant template from file

Parameters

`const char * filename`

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

`BYTE * isoTemplate`

[out] Pointer to memory space where the ISO/IEC 19794-2 compatible template will be loaded. Memory should be allocated before calling this function. The maximal size of ISO/IEC 19794-2 template is 1566 bytes.

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.
`IENGINE_E_BADTEMPLATE`	Input template is not valid ISO/IEC 19794-2 template.
`IENGINE_E_FILE`	Error occured while opening/reading file.

Related Topics

ISO_GetTemplateParameter (see page 27)

4.4.15 ISO_MergeTemplates Function

Combines finger views from two ISO/IEC 19794-2 templates into one common template

C++

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

Description

This function reads finger views from two ISO/IEC 19794-2 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 ISO/IEC 19794-2 template

`const BYTE * addedTemplate`

[in] ISO/IEC 19794-2 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 ISO/IEC 19794-2 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_OK`	No error occured.
`IENGINE_E_BADTEMPLATE`	At least one of the input templates is not valid ISO/IEC 19794-2 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

ISO_GetFingerView (see page 25)

4.4.16 ISO_RemoveMinutiae Function

Removes minutiae points from an ISO template

C++

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

Description

This function limits maximal number of minutiae points contained in an ISO/IEC 19794-2 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

`BYTE * inTemplate`

[in] Reference ISO/IEC 19794-2 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 ISO/IEC 19794-2 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_OK`	No error occured.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_MEMORY`	Memory allocation failed.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.

4.4.17 ISO_SaveTemplate Function

Stores ISO/IEC 19794-2 compliant template to file

C++

```
IENGINE_API int ISO_SaveTemplate(const char * filename, const BYTE * isoTemplate);
```

Description

This function stores the input ISO/IEC 19794-2 compliant template to a file

Parameters

`const char * filename`

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

`const BYTE * isoTemplate`

[in] ISO/IEC 19794-2 template

Return Values

Return Values	Description
`IENGINE_OK`	No error occured.
`IENGINE_E_INIT`	Library was not initialized.
`IENGINE_E_NULLPARAM`	NULL input parameter provided.
`IENGINE_E_BADTEMPLATE`	Input template is not valid ISO/IEC 19794-2 template.
`IENGINE_E_FILE`	Error occured while opening/reading file.

Related Topics

ISO_SetTemplateParameter (see page 30)

4.4.18 ISO_SetTemplateParameter Function

Set specific template parameter

C++

```
IENGINE_API int ISO_SetTemplateParameter(BYTE * isoTemplate, 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 ISO/IEC 19794-2 template. If specified template contains multiple finger views, this function modifies the first finger view (finger view with the lowest index number).

Parameters

`BYTE * isoTemplate`

[in/out] On input, this parameter should contain a valid ISO/IEC 19794-2 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_OK`	No error occured.
`IENGINE_E_BADPARAM`	Invalid parameter type provided.
`IENGINE_E_BADTEMPLATE`	Input template is not valid ISO/IEC 19794-2 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

ISO_GetTemplateParameter (see page 27)

5 Types, Structures, Enumerations

Enumerations

Name	Description
IENGINE_TEMPLATE_PARAMETER (see page 31)	Enumeration defining codes for parameters contained in ISO/IEC 19794-2 and ANSI/INCITS 378 compliant templates.
IENGINE_TEMPLATE_FORMAT (see page 32)	Enumeration defining codes for different template formats
IENGINE_FINGER_POSITION (see page 32)	Enumeration defining codes for different finger positions
IENGINE_IMPRESSION_TYPE (see page 32)	Defines impression type of fingerprint image
IENGINE_SORT_ORDER (see page 34)	Defines sort order of minutiae points

Structures

Name	Description
IENGINE_VERSION (see page 33)	Structure representing the current version of the library
IENGINE_MINUTIAE (see page 33)	Structure representing a particular minutia (distinctive fingerprint feature found in fingerprint skeleton, such as a bifurcation or an ending).

5.1 IENGINE_TEMPLATE_PARAMETER Enumeration

Enumeration defining codes for parameters contained in ISO/IEC 19794-2 and ANSI/INCITS 378 compliant templates.

C++

```
typedef enum {
  PARAM_PRODUCT_OWNER = 0,
  PARAM_PRODUCT_VERSION = 1,
  PARAM_TEMPLATE_SIZE = 2,
  PARAM_CAPTURE_EQUIPMENT_COMPLIANCE = 3,
  PARAM_CAPTURE_EQUIPMENT_ID = 4,
  PARAM_FINGER_VIEW_COUNT = 5,
  PARAM_FINGER_POSITION = 10,
  PARAM_IMPRESSION_TYPE = 11,
  PARAM_FINGER_QUALITY = 12
} IENGINE_TEMPLATE_PARAMETER;
```

Members

Members	Description
`PARAM_PRODUCT_OWNER = 0`	Specifies the 'owner' of the encoding application. This value is read-only (cannot be used with IEngine_SetTemplateParameter function).
`PARAM_PRODUCT_VERSION = 1`	Specifies the version of the encoding application. This value is read-only (cannot be used with IEngine_SetTemplateParameter function).
`PARAM_TEMPLATE_SIZE = 2`	Specifies the total length of the template in bytes. This value is read-only (cannot be used with IEngine_SetTemplateParameter function).
`PARAM_CAPTURE_EQUIPMENT_COMPLIANCE = 3`	Shall be a 4-bit value between 0 and 15, the most significant bit, if set to a 1, indicates that the equipment was cerified to comply with Appendix F (IAFIS Image Quality Specification, January 29, 1999) of FJIS-RS-0010, the Federal Bureau of Investigations's Electronic Fingerprint Transmission Specification. The other three bits are reserved for future compliance indicators. The default value for this parameter is 0.
`PARAM_CAPTURE_EQUIPMENT_ID = 4`	Shall be recorded in twelve bits. A value of all zeros are acceptable and idicate that the capture equipment ID is unreported. In other case, the value of the field is detemined by the vendor. The default value for this parameter is 0.
`PARAM_FINGER_VIEW_COUNT = 5`	Specifies total number of finger views contained within given template. This value is read-only (cannot be used with IEngine_SetTemplateParameter function).

PARAM_FINGER_POSITION = 10	Specifies the finger position of the encoded fingerprint. The values of different finger positions are defined in IENGINE_FINGER_POSITION (see page 32) enum. The default value for this parameter is 0 (UNKNOWN_FINGER).
PARAM_IMPRESSION_TYPE = 11	Specifies the impression type of the encoded fingerprint. The values of different finger positions are defined in IENGINE_IMPRESSION_TYPE (see page 32) enum. The default value for this parameter is 0 (TYPE_LIVE_SCAN_PLAIN).
PARAM_FINGER_QUALITY = 12	Specifies the quality of the encoded fingerprint. This quality number is an overall expression of the quality of the finger record, and represents quality of the original image. A value of 0 represents the lowest possible quality and the value 100 represents the highest possible quality. The numeric values in this field are set in accordance with the general guidelines contained in Section 2.1.42 of ANSI/INCITS 358. The default value for this parameter is 40 (fair quality).

5.2 IENGINE_TEMPLATE_FORMAT Enumeration

Enumeration defining codes for different template formats

C++

```
typedef enum {
  ANSI_TEMPLATE = 0,
  ISO_TEMPLATE = 1,
  ILO_SID_TEMPLATE = 2
} IENGINE_TEMPLATE_FORMAT;
```

5.3 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;
```

5.4 IENGINE_IMPRESSION_TYPE Enumeration

Defines impression type of fingerprint image

C++

```
typedef enum {
  TYPE_LIVE_SCAN_PLAIN = 0,
  TYPE_LIVE_SCAN_ROLLED = 1,
  TYPE_NONLIVE_SCAN_PLAIN = 2,
  TYPE_NONLIVE_SCAN_ROLLED = 3,
  TYPE_SWIPE = 4,
  TYPE_LIVE_SCAN_CONTACTLESS = 9
} IENGINE_IMPRESSION_TYPE;
```

5.5 IENGINE_VERSION Structure

Structure representing the current version of the library

C++

```
typedef struct iengine_version {
  unsigned int Major;
  unsigned int Minor;
} IENGINE_VERSION;
```

Description

Version definition contains major and minor number. The major number begins at 1 and is incremented by 1 for each major realease.

The minor number uses two digits to represent minor releases and revisions. The revision number is represented in the least significant digit. The more significant digits represent release minor version.

Example

Version	Major number	Minor number
1.0	1	0
1.01	1	1
1.11	1	11
1.24	1	24

5.6 iengine_version Structure

5.7 IENGINE_MINUTIAE Structure

Structure representing a particular minutia (distinctive fingerprint feature found in fingerprint skeleton, such as a bifurcation or an ending).

C++

```
typedef struct iengine_minutiae {
  BYTE angle;
  unsigned short x;
  unsigned short y;
  unsigned char type;
} IENGINE_MINUTIAE;
```

Members

Members	Description
`BYTE angle;`	Minutia angle encoded in one byte. Valid range: 0-255.
`unsigned short x;`	Minutiae x coordinate as stored in the template.

5

`unsigned short y;`	Minutiae y coordinate as stored in the template.
`unsigned char type;`	Minutiae type (bifurcation/ending)

5.8 IENGINE_SORT_ORDER Enumeration

Defines sort order of minutiae points

C++

```
typedef enum {
  SORT_NONE = 0,
  SORT_X_ASC = 1,
  SORT_X_DESC = 2,
  SORT_Y_ASC = 3,
  SORT_Y_DESC = 4
} IENGINE_SORT_ORDER;
```

Members

Members	Description
`SORT_NONE = 0`	No ordering required.
`SORT_X_ASC = 1`	Cartesian x-coordinate is used for ordering, ascending order.
`SORT_X_DESC = 2`	Cartesian x-coordinate is used for ordering, descending order.
`SORT_Y_ASC = 3`	Cartesian y-coordinate is used for ordering, ascending order.
`SORT_Y_DESC = 4`	Cartesian y-coordinate is used for ordering, descending order.

5.9 iengine_minutiae Structure

6 Constants

Macros

Name	Description
IENGINE_MIN_IMAGE_HEIGHT (see page 35)	Defines minimal height of accepted fingerprint images
IENGINE_MAX_IMAGE_HEIGHT (see page 35)	Defines maximal height of accepted fingerprint images
IENGINE_MIN_IMAGE_WIDTH (see page 35)	Defines minimal width of accepted fingerprint images
IENGINE_MAX_IMAGE_WIDTH (see page 35)	Defines maximal width of accepted fingerprint images
IENGINE_MAX_ANSI_TEMPLATE_SIZE (see page 36)	Maximal size of generated ANSI/INCITS 378 template (with only one finger view)
IENGINE_MAX_ISO_TEMPLATE_SIZE (see page 36)	Maximal size of generated ISO/IEC 19794-2 template (with only one finger view)

6.1 IENGINE_MIN_IMAGE_HEIGHT Macro

Defines minimal height of accepted fingerprint images

C++

```
#define IENGINE_MIN_IMAGE_HEIGHT 90
```

6.2 IENGINE_MAX_IMAGE_HEIGHT Macro

Defines maximal height of accepted fingerprint images

C++

```
#define IENGINE_MAX_IMAGE_HEIGHT 1800
```

6.3 IENGINE_MIN_IMAGE_WIDTH Macro

Defines minimal width of accepted fingerprint images

C++

```
#define IENGINE_MIN_IMAGE_WIDTH 90
```

6.4 IENGINE_MAX_IMAGE_WIDTH Macro

Defines maximal width of accepted fingerprint images

C++

```
#define IENGINE_MAX_IMAGE_WIDTH 1800
```

6.5 IENGINE_MAX_ANSI_TEMPLATE_SIZE Macro

Maximal size of generated ANSI/INCITS 378 template (with only one finger view)

C++

```
#define IENGINE_MAX_ANSI_TEMPLATE_SIZE 1568
```

6.6 IENGINE_MAX_ISO_TEMPLATE_SIZE Macro

Maximal size of generated ISO/IEC 19794-2 template (with only one finger view)

C++

```
#define IENGINE_MAX_ISO_TEMPLATE_SIZE 1566
```

7 Error Codes

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_FILE	1117	Error occured while opening/reading file.
IENGINE_E_INIT	1116	Library was not initialized.
IENGINE_E_MEMORY	1120	Memory allocation failed.
IENGINE_E_NULLPARAM	1121	NULL input parameter provided.
IENGINE_E_OTHER	1122	Other unspecified error.
IENGINE_E_BADFORMAT	1132	Unsupported format.
IENGINE_E_BADVALUE	1133	Invalid value provided.
IENGINE_E_BADLICENSE	1129	Provided license is not valid, or no license was found.
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)

8 Symbol Reference

8.1 Functions

The following table lists functions in this documentation.

8.2 Macros

The following table lists macros in this documentation.

8.3 Structs, Records, Enums

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

Index

Types, Structures, Enumerations 31