IBScanUltimate API Manual for C/C++

Version 2.0.1 (April 27, 2018)

Table of Contents

Revision History

Date	Author	Remarks
2018/4	YOUNG	Added descriptons for how to use main functions
2018/3	WADE	Added descriptions of new functions for IBScanUltimate v2.0.0.2b IBSU_AddFingerImage(), IBSU_RemoveFingerImage(), IBSU_IsFingerDuplicated(), IBSU_IsValidFingerGeometry()
2018/3	YOUNG	Added API function to improve dispaly speed on Embedded System (IBSU_GenerateDisplayImage)
2017/6	GON	Added descriptions of new functions for IBScanUltimate v1.9.6: IBSU_CheckWetFinger(), IBSU_GetImageWidth() and IBSU_IsWritableDirectory()
2017/4	GON	Added descriptions of new functions for IBScanUltimate v1.9.6: IBSU_CombineImageEx()
2015/12	YNG	Added descriptions of new functions for IBScanUltimate v1.9.0: IBSU_GetOperableBeeper(), IBSU_SetBeeper() Added descriptions of new callback function for IBScanUltimate v1.9.0: IBSU_CallbackKeyButton()
2015/8	YNG	Added descriptions of new functions for IBScanUltimate v1.8.5 IBSU_CombineImage()
2015/4	YNG	Added descriptions of new functions for IBScanUltimate v1.8.4: IBSU_UnloadLibrary() Added descriptions of exist callback functions
2015/3	YNG	Added descriptions of new functions for IBScanUltimate v1.8.3: IBSU_RedrawClientWindow() Changed descriptions of existing functions for IBScanUltimate v1.8.3: IBSU_WSQEncodeMem(), IBSU_ WSQEncodeToFile(), IBSU_ IBSU_WSQDecodeMem(), IBSU_ IBSU_WSQDecodeFromFile()
2014/09	YNG	Added descriptions of new functions for IBScanUltimate v1.8.1: IBSU_SavePngImage(), IBSU_SaveJP2Image()
2014/07	YNG	Added descriptions of new functions for IBScanUltimate v1.8.0: IBSU_WSQEncodeMem(), IBSU_WSQEncodeToFile(), IBSU_WSQDecodeMem(), IBSU_WSQDecodeFromFile(), IBSU_FreeMemory()
2013/10	BAN	Added descriptions of new functions for IBScanUltimate v1.7.0: IBSU_BGetImageEx(), IBSU_ReleaseCallbacks(), IBSU_SaveBitmapMem(), IBSU_ShowOverlayObject(), IBSU_ShowAllOverlayObject(), IBSU_RemoveOverlayObject(), IBSU_RemoveAllOverlayObject(), IBSU_AddOverlayText(), IBSU_ModifyOverlayText(), IBSU_AddOverlayLine(), IBSU_ModifyOverlayLine(), IBSU_AddOverlayQuadrangle(),

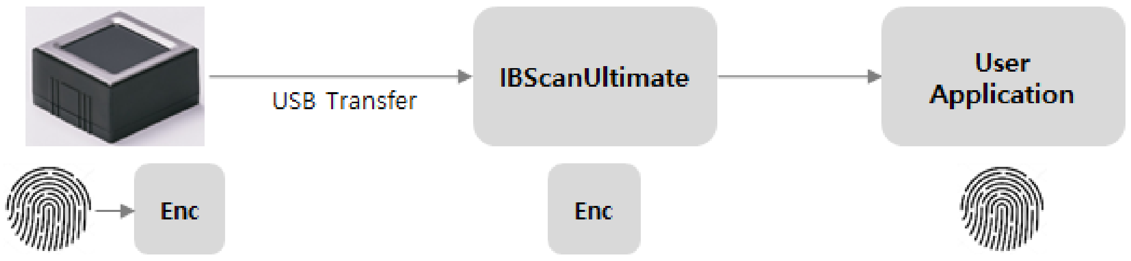
	IBSU_ModifyOverlayQuadrangle(), IBSU_AddOverlayShape(), IBSU_ModifyOverlayShape(). Noted that the ENUM_IBSU_EVENT_RESULT_IMAGE callback and IBSU_SetClientWindowOverlayText() are deprecated. Move client window functions into separate section.

1. How To

1.1. How to enable Encryption

- ***Description***

AES256-based encryption function gives user high safety of capture image.



However, it brings frame latency if Encrytion is enabled. So it is set to “Diable” as default.

- ***Available scanners and SDK versions***

Available in Wanson Mini. (Check available version with sales associate)

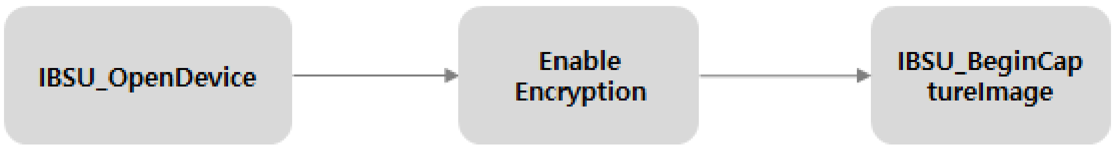
Available in SDK v1.10.x and later.

Available in all Operating Systems. (Windows, Linux, and Android)

- ***Usage***

1) Configuration

Call IBSU_SetProperty() with the property as below between the function call of “IBSU_OpenDevice”(or “IBSU_OpenDeviceEx”) and “IBSU_BeginCaptureImage”.



Function : IBSU_SetProperty

Property : ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION

2) Verification

Function : IBSU_GetProperty

Property : ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION

- ***Example***

1) Enable Encryption :

Between the function call of “IBSU_OpenDevice”(or “IBSU_OpenDeviceEx”) and “IBSU_CloseDevice”.

IBSU_SetProperty(devicehandle, ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION, "TRUE");

2) Disable Encryption :

IBSU_SetProperty(devicehandle, ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION, "FALSE");

3) Check status :

IBSU_GetProperty(devicehandle, ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION, szStatus);

case1) When Encryption is enabled

    szStatus = “TRUE”

case2) When Encryption is disabled

    szSatatus = “FALSE”

1.2. How to use IBScanNFIQ2

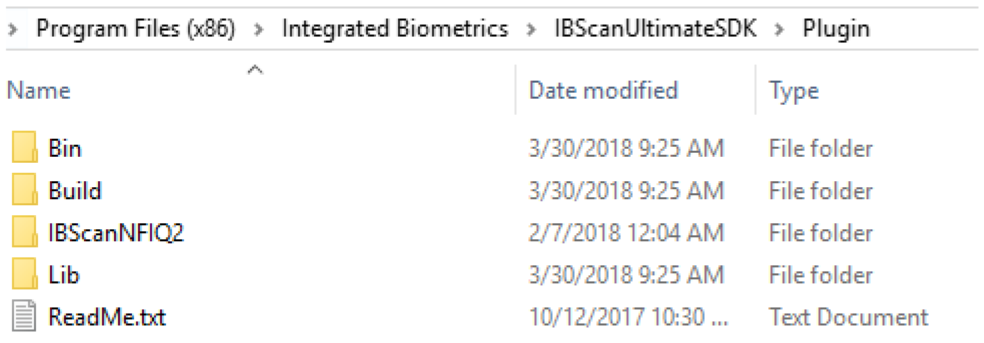
- ***Description***

IBScanNFIQ2 library was wrappered from NFIQ2.0 software developed by the National Institue of Standards and Technology (NIST).

Please refer to the more information at the link.

https://www.nist.gov/services-resources/software/development-nfiq-20

This library was included into Plugin folder as add-on. Please find the library and project samples from the folder “installed SDK\Plugin”



> Program Files (x86) > Integrated Biometrics > IBScanUltimateSDK > Plugin

Name	Date modified	Type
Bin	3/30/2018 9:25 AM	File folder
Build	3/30/2018 9:25 AM	File folder
IBScanNFIQ2	2/7/2018 12:04 AM	File folder
Lib	3/30/2018 9:25 AM	File folder
ReadMe.txt	10/12/2017 10:30 ...	Text Document

- ***Available scanners and SDK versions***

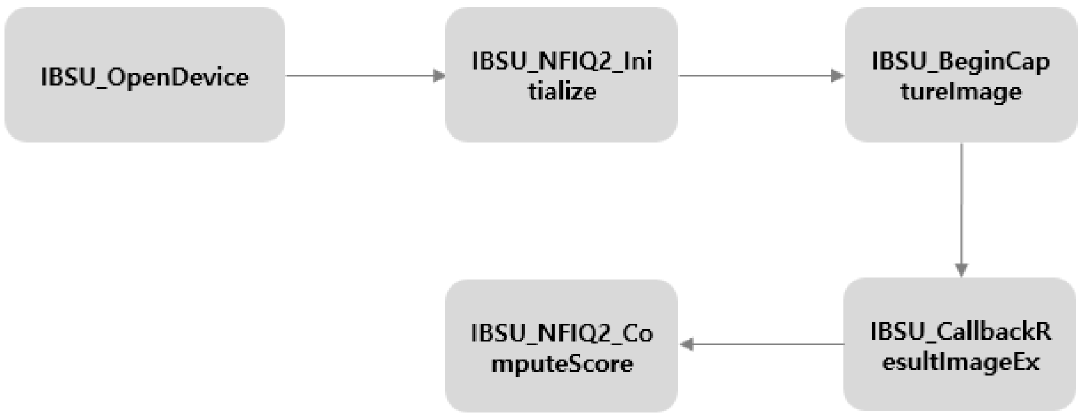
  Available in all IB scanners.

  Available in SDK v2.0.x and later.

  Available in Windows only.

- ***Usage***

  Call IBSU_NFIQ2_Initialize() and IBSU_NFIQ2_ComputeScore() between the function call of “IBSU_OpenDevice”(or “IBSU_OpenDeviceEx”) and “IBSU_CallbackResultImageEx callback”.



- ***Example***

  1) Initialize NFIQ2 :

  Initialize NFIQ2 module. It may takes few seconds depend on CPU

  After the function call of “IBSU_OpenDevice”(or “IBSU_OpenDeviceEx”).

IBSU_NFIQ2_Initialize(void);

2) Check if the NFIQ module is already initialzed :

IBSU_NFIQ2_IsInitizlized(void);

```
if (IBSU_NFIQ2_IsInitialized() != IBSU_STATUS_OK)
{
    // It may takes few seconds depend on CPU
    IBSU_NFIQ2_Initialize();
}
```

3) Compute NFIQ score :

IBSU_NFIQ2_ComputeScore(imgBuffer, width, height, bitsPerPixel, &pScore);

```
int nfiq_score2[IBSU_MAX_SEGMENT_COUNT];
int         score=0, nRc, segment_pos=0;
memset(&nfiq_score2, 0, sizeof(nfiq_score2));
for( int i=0; i<IBSU_MAX_SEGMENT_COUNT; i++ )
{
    if( pDlg->m_FingerQuality[i] == ENUM_IBSU_FINGER_NOT_PRESENT )
        continue;

    nRc = IBSU_NFIQ2_ComputeScore((const BYTE*)(pSegmentImageArray+segment_pos)-
>Buffer, (pSegmentImageArray+segment_pos)->Width, pSegmentImageArray+segment_pos)-
>Height, (pSegmentImageArray+segment_pos)->BitsPerPixel, &score);
    if( nRc == IBSU_STATUS_OK )
        nfiq_score2[i] = score;
    else
        nfiq_score2[i] = -1;
    segment_pos++;
}
```

1.3. How to use Duplicate Finger

- ***Description***

Through Duplicate Finger which IBScanUltimate supports, user can identify fingers. User needs to register fingers first, and match fingers with the registered fingers.

Fingers are used as special features which are extracted by IB extraction algorithm. These extractions are used when they are saved to the buffer of IBScanUltimate, and when user tries to match input-fingers with them.

1) IBSU_AddFingerImage : Registers fingerimages to the designated position

2) IBSU_RemoveFingerImage : Unregisters fingerimages from the designated position

3) IBSU_IsFingerDuplicated : Matches fingerimages with the designated positions and returns the result.

- ***Available scanners and SDK versions***

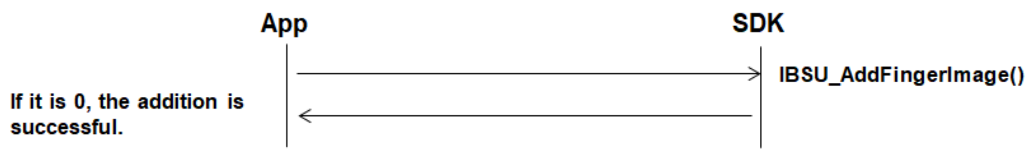
Available in all IB Fingerprint scanners.

Available in SDK v2.0.1 and later.

Available in all Operating Systems. (Windows, Linux, and Android)

- ***Usage***

1) Register finger

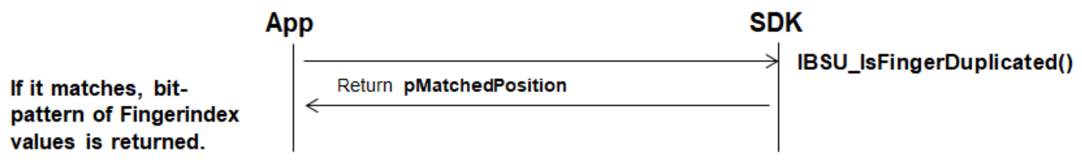


Function : IBSU_AddFingerImage

Parameters : Refers to "1.1.3.40) IBSU_ AddFingerImage"

Users should designate positions of buffer for finger images to be saved. The positions are defined with bit-patterns in "IBScanUltimateApi_Def.h".

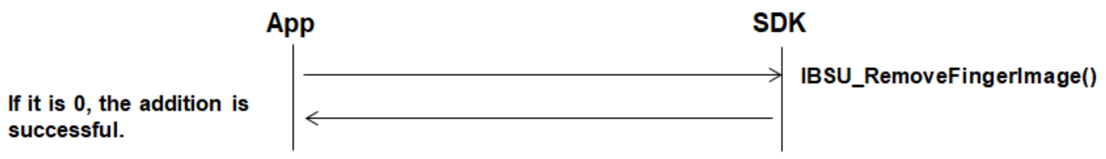
2) Identify finger


App
SDK
IBSU_IsFingerDuplicated()
Return pMatchedPosition
If it matches, bit-pattern of Fingerindex values is returned.

Function : IBSU_IsFingerDuplicated

Property : Refers to "1.1.3.42) IBSU_IsFingerDuplicated"

3) Un-Register finger



Function : IBSU_RemoveFingerImage

Parameters : Refers to "1.1.3.41) IBSU_ RemoveFingerImage"

- ***Example***

  1) Register finger

  [CASE] When user registers R-Index finger

  IBSU_AddFingerImage(deviceHandle, image, IBSU_FINGER_RIGHT_INDEX, ENUM_IBSU_FLAT_SINGLE_FINGER, FALSE);

  2) Identify finger

  [Case 1] Not matched

  R-Index Finger is registered, but R-Middle Finger is captured and call as below.

  IBSU_IsFingerDuplicated(deviceHandle, image, IBSU_FINGER_RIGHT_INDEX, ENUM_IBSU_FLAT_SINGLE_FINGER, 4, pMatchedPosition);

  "pMatchedPosition" is returned with '0'.

  [Case 2] Matched

  R-Index Finger is registered, but R-Index Finger is captured and call as below.

  IBSU_IsFingerDuplicated(deviceHandle, image, IBSU_FINGER_RIGHT_INDEX, ENUM_IBSU_FLAT_SINGLE_FINGER, 4, pMatchedPosition);

  "pMatchedPosition" is returned with 'IBSU_FINGER_RIGHT_INDEX'.

  3) Un-Register finger

  [Case] When user removes R-Index finger

  IBSU_RemoveFingerImage(deviceHandle, IBSU_FINGER_RIGHT_INDEX);

  4) Update finger

  [Case] When user updates R-Index finger.

  IBSU_AddFingerImage(deviceHandle, image, IBSU_FINGER_RIGHT_INDEX, ENUM_IBSU_FLAT_SINGLE_FINGER, ***TRUE***);

1.4. How to use Hand Checker

- ***Description***

IBScanUltimate supports Hand checker which identifies fingers are in right places.

It is valid on the identification for 4-finger and 2-finger. For example, In case of 4-finger it can identify left or right hand, and in case of 2-finger it can identify “little-ring” or “index-middle”.

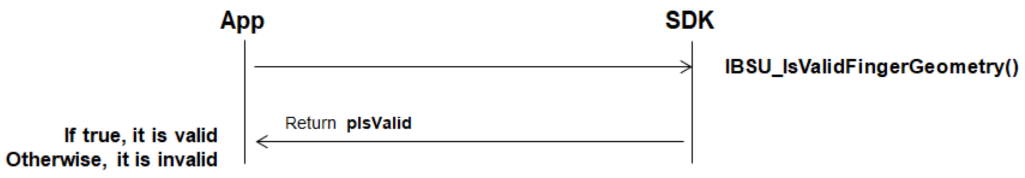
If it matches, “TRUE” is returned in Boolean type

.

- ***Available scanners and SDK versions***

Available in 2-finger and 4-finger Fingerprint scanners.
Available in SDK v2.0.1 and later.
Available in all Operating Systems. (Windows, Linux, and Android)

- ***Usage***



Function : IBSU_IsValidFingerGeometry

Parameters : Refers to “1.1.3.43) IBSU_IsValidFingerGeometry”

- ***Example***

1) Hand Check(4-finger)
[CASE 1] Matched
“Right hand” check when captured the Right 4 fingers

```
IBSU_IsValidFingerGeometry(deviceHandle, image, IBSU_FINGER_RIGHT_HAND, ENUM_IBSU_FLAT_FOUR_FINGERS, &isValid);
```

“TRUE” is returned to “isValid” in Boolean type

[CASE 2] Not Matched
“Right hand” check when captured the left 4 fingers

```
IBSU_IsValidFingerGeometry(deviceHandle, image, IBSU_FINGER_RIGHT_HAND, ENUM_IBSU_FLAT_FOUR_FINGERS, &isValid);
```

“FALSE” is returned to “isValid” in Boolean type

2) 2-finger Check
[CASE 1] Matched
“R-Index and R-Middle finger” check when captured the Right index and middle fingers

```
IBSU_IsValidFingerGeometry(deviceHandle, image, IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE,
                           ENUM_IBSU_FLAT_TWO_FINGERS, &isValid);
```

"TRUE" is returned to "isValid" in Boolean type

[CASE 2] Not Matched

"R-Index and R-Middle finger" check when captured the Right ring and little fingers

```
IBSU_IsValidFingerGeometry(deviceHandle, image, IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE,
                           ENUM_IBSU_FLAT_TWO_FINGERS, &isValid);
```

"FALSE" is returned to "isValid" in Boolean type

2. API Function Lists

2.1. C Functions

2.1.1. Summary of All Interface Functions

No	Functions
Main Interface Functions	
1	int WINAPI IBSU_GetSDKVersion (IBSU_SdkVersion *pVerinfo)
2	int WINAPI IBSU_GetDeviceCount (int *pDeviceCount)
3	int WINAPI IBSU_GetDeviceDescription (const int deviceIndex, IBSU_DeviceDesc *pDeviceDesc)
4	int WINAPI IBSU_RegisterCallbacks (const int handle, const IBSU_Events events, void *pEventName, void *pContext)
5	int WINAPI IBSU_ReleaseCallbacks(const int handle, const IBSU_Events events)
6	Int WINAPI IBSU_OpenDevice (const int deviceIndex, int *pHandle)
7	int WINAPI IBSU_CloseDevice (const int handle)
8	Int WINAPI IBSU_CloseAllDevice()
9	int WINAPI IBSU_IsDeviceOpened (const int handle)
10	int WINAPI IBSU_SetProperty(const int handle, const IBSU_PropertyId propertyId, LPCSTR propertyValue)
11	int WINAPI IBSU_GetProperty(const int handle, const IBSU_PropertyId propertyId, LPSTR propertyValue)
12	int WINAPI IBSU_AsyncOpenDevice(const int deviceIndex)
13	Int WINAPI IBSU_OpenDeviceEx(const int deviceIndex, LPCSTR uniformityMaskPath, const BOOL asyncOpen, int *pHandle)
14	int WINAPI IBSU_EnableTraceLog(BOOL on)
15	int WINAPI IBSU_UnloadLibrary()
Image Acquisition Related Interface Functions	
1	int WINAPI IBSU_IsCaptureAvailable (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, BOOL *pIsAvailable)
2	int WINAPI IBSU_BeginCaptureImage (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, const DWORD captureOptions)
3	int WINAPI IBSU_CancelCaptureImage (const int handle)
4	int WINAPI IBSU_IsCaptureActive (const int handle, BOOL *pIsActive)
5	int WINAPI IBSU_TakeResultImageManually(const int handle)
6	int WINAPI IBSU_GetContrast (const int handle, int *pContrastValue)
7	int WINAPI IBSU_SetContrast (const int handle, const int contrastValue)
8	Int WINAPI IBSU_SetLEOperationMode(const int handle, const IBSU_LEOperationMode leOperationMode)

9	Int WINAPI IBSU_GetLEOperationMode(const int handle, IBSU_LEOperationMode *leOperationMode)
10	int WINAPI IBSU_IsTouchedFinger (const int handle, int *touchInValue)
11	Int WINAPI IBSU_GetOperableLEDs (const int handle, IBSU_LedType *pLedType, int *pLedCount, DWORD *pOperableLEDs)
12	Int WINAPI IBSU_GetLEDs (const int handle, DWORD *pActiveLEDs)
13	Int WINAPI IBSU_SetLEDs (const int handle, const DWORD activeLEDs)
14	Int WINAPI IBSU_GenerateZoomOutImage (const IBSU_ImageData inImage, BYTE *outImage, const int outWidth, const int outHeight, const BYTE bkColor)
15	int WINAPI IBSU_SaveBitmapMem(const BYTE *inImage, const DWORD inWidth, const DWORD inHeight, const int inPitch, const double inResX, const double inResY, BYTE *outBitmapBuffer, const IBSU_ImageFormat outImageFormat, const DWORD outWidth, const DWORD outHeight, const BYTE bkColor)
16	Int WINAPI IBSU_SaveBitmapImage (LPCSTR filepath, const BYTE *imgBuffer, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)
17	int WINAPI IBSU_BGetImage(const int handle, IBSU_ImageData *pImage, IBSU_ImageType *pImageType, IBSU_ImageData *pSplitImageArray, int *pSplitImageArrayCount, IBSU_FingerCountState *pFingerCountState, IBSU_FingerQualityState *pQualityArray, int *pQualityArrayCount)
18	int WINAPI IBSU_BGetImageEx(const int handle, int *pImageStatus, IBSU_ImageData *pImage, IBSU_ImageType *pImageType, int *pDetectedFingerCount, IBSU_ImageData *pSegmentImageArray, IBSU_SegmentPosition *pSegmentPositionArray, int *pSegmentImageArrayCount, IBSU_FingerCountState *pFingerCountState, IBSU_FingerQualityState *pQualityArray, int *pQualityArrayCount)
19	int WINAPI IBSU_BGetInitProgress(const int deviceIndex, BOOL *pIsComplete, int *pHandle, int *pProgressValue)
20	int WINAPI IBSU_BGetClearPlatenAtCapture(const int handle, IBSU_PlatenState *pPlatenState)
21	Int WINAPI IBSU_BGetRollingInfo(const int handle, IBSU_RollingState *pRollingState, int *pRollingLineX)
22	Int WINAPI IBSU_GetIBSM_ResultImageInfo(const int handle, IBSM_FingerPosition fingerPosition, IBSM_ImageData *pResultImage, IBSM_ImageData *pSplitResultImageCount)
23	Int WINAPI IBSU_GetNFIQScore(const int handle, const BYTE *imgBuffer, const DWORD width, const DWORD height, const BYTE bitsPerPixel, int *pScore)
24	Int WINAPI IBSU_GenerateZoomOutImageEx(const BYTE *pInImage, const int inWidth, const int inHeight, BYTE *outImage, const int outWidth, const int outHeight, const BYTE bkColor)
25	Int WINAPI IBSU_WSQEncodeMem(const BYTE *image, const int width, const int height, cons tint pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText, BYTE **compressed Data, int *compressedLength)

26	Int WINAPI IBSU_WSQEncodeToFile(LPCSTR filePath, const BYTE *image, const int width, const int height, cons tint pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText)
27	Int WINAPI IBSU_WSQDecodeMem(const BYTE *compressedImage, const int compressedLength, BYTE **decompressedImage, int *outWidth, int *outHeight, int *outPitch, int *outBitsPerPixel, int *outPixelPerInch)
28	Int WINAPI IBSU_WSQDecodeFromFile(LPCSTR filePath, BYTE **decompressedImage, int *outWidth, int *outHeight, int *outpitch, int *outBitsPerPixel, int *outPixelPerInch)
29	Int WINAPI IBSU_FreeMemory(void *memblock)
30	Int WINAPI IBSU_SavePngImage (LPCSTR filepath, const BYTE *image, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)
31	Int WINAPI IBSU_SaveJP2Image (LPCSTR filepath, const BYTE *image, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY, const int fQuality)
32	Int WINAPI IBSU_CombineImage (const IBSU_ImageData inImage1, const IBSU_ImageData inImage2,IBSU_CombineImageWhichHand whichHand , IBSU_ImageData *ouImage)
33	Int WINAPI IBSU_GetOperableBeeper (const int handle, IBSU_BeeperType *pBeeperType)
34	Int WINAPI IBSU_SetBeeper (const int handle, const IBSU_BeepPattern beepPattern, const DWORD soundTone, const DWORD duration, const DWORD reserved_1, const DWORD reserved_2)
35	Int WINAPI IBSU_CombineImageEx (const IBSU_ImageData inImage1, const IBSU_ImageData inImage2,IBSU_CombineImageWhichHand whichHand , IBSU_ImageData *ouImage, IBSU_ImageData *pSegmentImageArray, IBSU_SegmentPosition *pSegmentPositionArray, int *pSegmentImageArrayCount)
36	Int WINAPI IBSU_CheckWetFinger (const int handle, const IBSU_ImageData inImage)
37	Int WINAPI IBSU_GetImageWidth (const int handle, const IBSU_ImageData inImage, int *Width_MM)
38	Int WINAPI IBSU_IsWritableDirectory (LPCSTR filepath, BOOL needCreateSubFolder)
39	int WINAPI IBSU_GenerateDisplayImage(const BYTE *pInImage, const int inWidth, const int inHeight, BYTE *outImage, const int outWidth, const int outHeight, const BYTE outBkColor, const IBSU_ImageFormat outFormat, const int outQualityLevel, const BOOL outVerticalFlip)
40	int WINAPI IBSU_AddFingerImage(const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const BOOL flagForce)
41	int WINAPI IBSU_RemoveFingerImage(const int handle, const DWORD fIndex)
42	int WINAPI IBSU_IsFingerDuplicated(const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const int securityLevel, BOOL *pDuplicated)

43	int WINAPI IBSU_IsValidFingerGeometry(const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, BOOL *pValid)
Client Window Functions	
1*	int WINAPI IBSU_CreateClientWindow (const int handle, const IBSU_HWD hWindow, const DWORD left, const DWORD top, const DWORD right, const DWORD bottom)
2*	Int WINAPI IBSU_DestroyClientWindow(const int handle, const BOOL clearExistingInfo)
3*	Int WINAPI IBSU_GetClientWindowProperty(const int handle, const IBSU_ClientWindowPropertyId propertyId, LPSTR propertyValue)
4*	int WINAPI IBSU_SetClientWindowProperty (const int handle, const IBSU_ClientWindowPropertyId propertyId, LPCSTR propertyValue)
5*	Int WINAPI IBSU_SetClientWindowOverlayText (const int handle, const char *fontName, const int fontSize, const BOOL fontBold, const char *text, const int posX, cons tint posY, const DWRD textColor)
6*	int WINAPI IBSU_ShowOverlayObject(const int handle, const int overlayHandle, const BOOL show);
7*	int WINAPI IBSU_ShowAllOverlayObject(const int handle, const BOOL show);
8*	int WINAPI IBSU_RemoveOverlayObject(const int handle, const int overlayHandle);
9*	int WINAPI IBSU_RemoveAllOverlayObject(const int handle);
10*	int WINAPI IBSU_AddOverlayText(const int handle, int *pOverlayHandle, const char *fontName, const int fontSize, const BOOL fontBold, const char *text, const int posX, const int posY, const DWORD textColor);
11*	int WINAPI IBSU_ModifyOverlayText(const int handle, const int overlayHandle, const char *fontName, const int fontSize, const BOOL fontBold, const char *text, const int posX, const int posY, const DWORD textColor);
12*	int WINAPI IBSU_AddOverlayLine(const int handle, int *pOverlayHandle, const int x1, const int y1, const int x2, const int y2, const int lineWidth, const DWORD lineColor);
13*	int WINAPI IBSU_ModifyOverlayLine(const int handle, const int overlayHandle, const int x1, const int y1, const int x2, const int y2, const int lineWidth, const DWORD lineColor);
14*	int WINAPI IBSU_AddOverlayQuadrangle(const int handle, int *pOverlayHandle, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3, const int x4, const int y4, const int lineWidth, const DWORD lineColor);
15*	int WINAPI IBSU_ModifyOverlayQuadrangle(const int handle, const int overlayHandle, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3, const int x4, const int y4, const int lineWidth, const DWORD lineColor);
16*	int WINAPI IBSU_AddOverlayShape(const int handle, int *pOverlayHandle, const IBSU_OverlayShapePattern shapePattern, const int x1, const int y1, const int x2, const int y2, const int lineWidth, const DWORD lineColor, const

	int reserved_1, const int reserved_2);
17*	int WINAPI IBSU_ModifyOverlayShape(const int handle, const int overlayHandle, const IBSU_OverlayShapePattern shapePattern, const int x1, const int y1, const int x2, const int y2, const int lineWidth, const DWORD lineColor, const int reserved_1, const int reserved_2);
18	Int WINAPI IBSU_RedrawClientWindow(cons tint handle);
Callback Interface Functions	
1	typede void (CALLBACK *IBSU_Callback) (const int deviceHandle, void *pContext)
2	typede void (CALLBACK *IBSU_CallbackPreviewImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image)
3	typede void (CALLBACK *IBSU_CallbackFingerCount) (const int deviceHandle, void *pContext, const IBSU_FingerCountState fingerCountState)
4	typede void (CALLBACK *IBSU_CallbackFingerQuality) (const int deviceHandle, void *pContext, const IBSU_FingerQualityState *pQualityArray, const int qualityArrayCount)
5	typede void (CALLBACK *IBSU_CallbackDeviceCount) (const int detectedDevices, void *pContext)
6	typede void (CALLBACK *IBSU_CallbackInitProgress) (const int deviceIndex, void *pContext, const int progressValue)
7	typede void (CALLBACK *IBSU_CallbackTakingAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)
8	typede void (CALLBACK *IBSU_CallbackCompleteAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)
9	typede void (CALLBACK *IBSU_CallbackResultImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image, const IBSU_ImageType imageType, const IBSU_ImageData *pSplitImageArray, const int splitImageArrayCount)
10	typede void (CALLBACK *IBSU_CallbackResultImageEx) (const int deviceHandle, void *pContext, const int imageStatus, const IBSU_ImageData image, const IBSU_ImageType imageType, const int detectedFingerCount, const int segmentImageArrayCount, const IBSU_ImageData *pSegmentImageArray, const IBSU_SegmentPosition *pSegmentPositionArray)
11	typede void (CALLBACK *IBSU_CallbackClearPlatenAtCapture) (const int deviceHandle, void *pContext, const IBSU_PlatenState platenState)
12	typede void (CALLBACK *IBSU_CallbackAsyncOpenDevice) (const int deviceIndex, void *pContext, const int deviceHandle, const int errorCode)
13	typede void (CALLBACK *IBSU_CallbackNotifyMessage)

	(const int deviceHandle, void *pContext, const int notifyMessage)
14	typede void (CALLBACK *IBSU_CallbackKeyButtons) (const int deviceHandle, void *pContext, const int pressedKeyButtons)

Table 1
*** Available only on Windows**

2.1.2.Main Interface Functions

1.1.2.1) IBSU_GetSDKVersion

- ***Prototype***

API DLL	int WINAPI IBSU_GetSDKVersion (IBSU_SdkVersion *pVerinfo)

- ***Description***

  Gets a structure holding product and software version information (IBSU_SdkVersion).

- ***Parameter***

Parameter	Description
*pVerinfo	***[out] API version information. Memory must be provided by caller.***

- ***IBSU_SdkVersion Structure Definition***

```
/* Container to hold version information. */
#define IBSU_MAX_STR_LEN 128

typedef struct tagIBSU_SdkVersion
{
    char Product[IBSU_MAX_STR_LEN];                    /* Product version string. */
    char File[IBSU_MAX_STR_LEN];              /* File version string. */
}
```

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.2.2) IBSU_GetDeviceCount

- ***Prototype***

API DLL	int WINAPI IBSU_GetDeviceCount (int *pDeviceCount)

- ***Description***

  Retrieves count of connected IB USB Scanner devices.

- ***Parameter***

Parameter	Description
*pDeviceCount	***[out] Number of connected devices. Memory must be provided by caller.***

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
int devices = 0;
 IBSU_GetDeviceCount( &devices );
```

1.1.2.3) IBSU_GetDeviceDescription

- ***Prototype***

API DLL	int WINAPI IBSU_GetDeviceDescription (const int deviceIndex, IBSU_DeviceDesc *pDeviceDesc)

- ***Description***

  Retrieves detailed device information about a particular scanner by logical index.

- ***Parameter***

Parameter	Description
deviceIndex	[in] Zero-based device index for device to lookup
*pDeviceDesc	***[out] Basic device information. Memory must be provided by caller.***

- ***IBSU_DeviceDescription Structure Definition***

```
typedef struct tagIBSU_DeviceDesc
{
    char serialNumber[128];   /* Device serial number */
    char productName[128];    /* Device product name */
    char interfaceType[128];  /* Device interface type (USB, Firewire) */
    char fwVersion[128];      /* Device firmware version */
    char  devRevision[128];   /*  Device  revision */
    int   handle;             /*  Return device handle */
    bool IsHandleOpened;      /*  Check  if  device handle is opened */
    #ifdef __android__
    int   devID;                         /* Device ID. */
    #endif
}
```

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
IBSU_DeviceDesc devDesc;
devDesc.productName[0] = 0;
if( deviceIndex > -1 )
        IBSU_GetDeviceDescription( deviceIndex, &devDesc );
```

1.1.2.4) IBSU_RegisterCallbacks

- ***Prototype***

API DLL	int WINAPI IBSU_RegisterCallbacks (const int handle, const IBSU_Events events, void *pEventName, void *pContext)

- ***Description***

This function is used to register callback methods, utilizing event-driven programming when the state of the scanner changes.

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
events	[in] Event for which callback function is being registered.
*pEventName	[in] Pointer to the notification function
*pContext	[in] Pointer to user context; this value is used as parameter for callback

- ***IBSU_Events Enumerations***

```
/* Callback when device count changes. */
ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT
/* Callback when communication with a device is interrupted. */
ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK
```

```
/* Callback when communication with a device is interrupted. */
ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE
/* Callback for rolled print acquisition when rolling should begin. */
ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION
/* Callback for rolled print acquisition when rolling completes. */
ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION
/* Callback when result image is available for a capture (deprecated). */
ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE
/* Callback when a finger quality changes. */
ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT
/* Callback when the finger count changes. */
ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS
/* Callback when initialization progress changes for a device. */
ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE
/* Callback when asynchronous device initialization completes. */
ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE
/* Callback when a warning message is generated. */
ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE
/* Callback when result image is available for a capture (with extended information). */
ENUM_IBSU_ESSENTIAL_EVEN_RESULT_IMAGE_EX
```

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Examples***

```
IBSU_RegisterCallbacks( deviceHandle,
          ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK,
          pDialog->OnDeviceComunicationBreak, pDialog );

IBSU_RegisterCallbacks( deviceHandle,
          ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE,
          pDialog->OnPreviewImageAvailable, pDialog );

IBSU_RegisterCallbacks( deviceHandle,
          ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT,
                pDialog->OnFingerCountChange, pDialog );

        IBSU_RegisterCallbacks( deviceHandle,
                ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION,
                pDialog->OnTakingAcquisition, pDialog );

        IBSU_RegisterCallbacks( deviceHandle,
                ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION,
                pDialog->OnCompleteAcquisition, pDialog );

        IBSU_RegisterCallbacks( deviceHandle,
            ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE,
            pDialog->OnResultImageAvailable, pDialog );

        IBSU_RegisterCallbacks( NULL,
            ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS,
            pDialog->OnInitProgressChange, pDialog );

        IBSU_RegisterCallbacks(deviceHandle,
            ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE,
            pDialog->OnClearPlatenAtCapture, pDialog );

        IBSU_RegisterCallbacks( NULL,
            ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE,
            pDialog->OnAsyncOpenDevice, pDialog );

        IBSU_RegisterCallbacks(deviceHandle,
            ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE,
            pDialog->OnNotifyMessage, pDialog );

        IBSU_RegisterCallbacks( deviceHandle,
            ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE_EX,
            pDialog->OnResultImageAvailableEx, pDialog );
```

1.1.2.5) IBSU_ReleaseCallbacks

- ***Prototype***

API DLL	int WINAPI IBSU_ReleaseCallbacks(const int handle, const IBSU_Events events);

- ***Description***

  Unregister a callback function for a particular event.

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
events	[in] Event for which the callback is being unregistered.

- ***IBSU_Events Enumerations***

/* Callback when device count changes. */

ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT

/* Callback when communication with a device is interrupted. */

ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK

/* Callback when communication with a device is interrupted. */

ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE

/* Callback for rolled print acquisition when rolling should begin. */

ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION

/* Callback for rolled print acquisition when rolling completes. */

ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION

/* Callback when result image is available for a capture (deprecated). */

ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE

/* Callback when a finger quality changes. */

ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT

/* Callback when the finger count changes. */

ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS

/* Callback when initialization progress changes for a device. */

ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE

/* Callback when asynchronous device initialization completes. */

ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE

/* Callback when a warning message is generated. */

ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE

/* Callback when result image is available for a capture (with extended information). */

ENUM_IBSU_ESSENTIAL_EVEN_RESULT_IMAGE_EX

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.2.6) IBSU_OpenDevice

- ***Prototype***

API DLL	int WINAPI IBSU_OpenDevice (const int deviceIndex, int *pHandle)

- ***Description***

Initializes a device, given a particular device index.

- ***Parameter***

Parameter	Description
deviceIndex	[in] Zero-based device index for device to init
*pHandle	***[out] Function returns device handle to be used for subsequent function calls. Memory must be provided by caller.***

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
nRc = IBSU_OpenDevice( deviceIndex, &deviceHandle );
if( nRc >= IBSU_STATUS_OK )
{
        // deviceHandle will be needed for subsequent calls to device.
        pDialog->m_DeviceHandle = deviceHandle;
}
```

- ***Note***

Any initialized device must be released before closing the host application! (call IBSU_CloseDevice() or IBSU_CloseAlldevice())

1.1.2.7) IBSU_CloseDevice

- ***Prototype***

API DLL	int WINAPI IBSU_CloseDevice (const int handle)

- ***Description***

Releases a device (by device handle).

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h IBSU_ERR_RESOURCE_LOCKED -> a callback is still active IBSU_ERR_NOT_INITIALIZED-> device(s) in use are identified by index; so either device has already been released or is unknown

- ***Example***

```
int nRc = IBSU_STATUS_OK;
if( m_DeviceHandle != -1 )
nRc = IBSU_CloseDevice( m_DeviceHandle );
if( nRc >= IBSU_STATUS_OK )
{
        m_DeviceHandle = -1;
        m_IsInitializing = false;
}
```

1.1.2.8) IBSU_ CloseAllDevice

- ***Prototype***

API DLL	int WINAPI IBSU_CloseAllDevice()

- ***Description***

Releases all currently initialized devices (particular device handle not needed).

- ***Parameter***

Parameter	Description

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *IBSU_ERR_RESOURCE_LOCKED -> a callback is still active*

- ***Note***

  This function should be called upon closing the host application to free allocated resources.

1.1.2.9) IBSU_IsDeviceOpened

- ***Prototype***

API DLL	int IBSU_IsDeviceOpened (const int handle)

- ***Description***

  Check if a particular device is opened/initialized.

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h IBSU_STATUS_OK -> device is ready to be used IBSU_ERR_INVALID_PARAM_VALUE -> if handle value is out of valid range IBSU_ERR_NOT_INITIALIZED -> device is not initialized IBSU_ERR_DEVICE_IO -> device is initialized but there was a communication problem

1.1.2.10) IBSU_SetProperty

- ***Prototype***

API DLL	int IBSU_SetProperty(const int handle, const IBSU_PropertyId propertyId, LPCSTR propertyValue)

- ***Description***

Set a device’s property value (by handle).

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
propertyId	[in] Property identifier to set value for.
propertyValue	[in] String containing property value.

- ***IBSU_PropertyId Enumerations (Settable)***

/* Time to acquire fingerprint in the auto capture regardless of number of fingers. The option IBSU_OPTION_AUTO_CAPTURE must be used. The valid range is between 2000- and 4000-ms, inclusive, with the default of 4000-ms. [Get and set.] */

ENUM_IBSU_PROPERTY_IGNORE_FINGER_TIME

/* Auto contrast level value. [Get and set.] */

ENUM_IBSU_PROPERTY_RECOMMENDED_LEVEL

/* Enable power save mode (TRUE to enable or FALSE to disable). [Get and set.] */

ENUM_IBSU_PROPERTY_ENABLE_POWER_SAVE_MODE

/* Retry count for communication error. The valid range is between 0 and 120, inclusi ve, with the default of 6. [Get and set.] */

ENUM_IBSU_PROPERTY_RETRY_WRONG_COMMUNICATION

/* The maximum wait time for image capture, in seconds. Must use IBSU_CallbackRe sultImageEx instead of IBSU_CallbackResultImage. If -1, the timeout is infinite. Other wise, the valid range is between 10- and 3600-seconds, inclusive. The default is -1. [Get and set.] */

ENUM_IBSU_PROPERTY_CAPTURE_TIMEOUT

/* Minimum distance of rolled fingerprint, in millimeters. The valid range is between 1 0- and 30-mm, inclusive. The default is 15-mm. [Get and set.] */

ENUM_IBSU_PROPERTY_ROLL_MIN_WIDTH

/* roll mode. The valid range is between 0 ~ 1. The default is 1. [Get and set.]

0 : no use smear

1 : use notice */

ENUM_IBSU_PROPERTY_ROLL_MODE,

/* roll level. The valid range is between 0 ~ 2. The default is 1. [Get and set.]

0 : low level

1 : medium level

2 : high level */

ENUM_IBSU_PROPERTY_ROLL_LEVEL,

/* The area threshold for image capture for flat fingers. The area threshold for beginning rolled finger. The valid range is between 0 and 12, inclusive, with the default of 6. [Get and set.] */

ENUM_IBSU_PROPERTY_CAPTURE_AREA_THRESHOLD,

/* Enable decimation mode (TRUE to enable or FALSE to disable). Some of devices (or firmware version) does not support this feature. [Get and set.]*/

ENUM_IBSU_PROPERTY_ENABLE_DECIMATION,

/* Enable capture on release (TRUE to enable or FALSE to disable). The default is FALSE. [Get and set.]

TRUE: the result callback will be called when user release the finger from the sensor.

FALSE: the result callback will be called when the quality of finger become good */

ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_ON_RELEASE,

/* It can be used for dry finger. Some of devices (or firmware version) does not support this feature. The default is FALSE. [Get and set.]

TRUE : Enable dry mode.

FALSE : Disable dry mode */

ENUM_IBSU_PROPERTY_SUPER_DRY_MODE,

/* It is a minimum capture time when the dry mode is enabled with the property ENUM_IBSU_PROPERTY_SUPER_DRY_MODE. Some of devices (or firmware version) does not support this feature. The valid range is between 600- and 3000-ms, inclusive, with the default of 2000-ms. [Get and set.]*/

ENUM_IBSU_PROPERTY_MIN_CAPTURE_TIME_IN_SUPER_DRY_MODE,

/* Enable the drawing for preview image (TRUE to enable or FALSE to disable).
The default is TRUE. [Get and set.] */

ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE,

/* Enable to override roll image (TRUE to enable or FALSE to disable).
The default is FALSE. [Get and set.] */

ENUM_IBSU_PROPERTY_ROLL_IMAGE_OVERRIDE,

/* Enable the warning message for invalid area for result image (TRUE to enable or FALSE to disable).

```
  The default is FALSE.  [Get and set.] */
ENUM_IBSU_PROPERTY_WARNING_MESSAGE_INVALID_AREA,
/* Enable wet detect function.
  The default is FALSE.  [Get and set.] */
ENUM_IBSU_PROPERTY_ENABLE_WET_FINGER_DETECT,
/* Change wet detect level.
  * The valid range is between 1 and 5. The default is 3.  [Get and set.]
  * 1 : Lowest level for detect wet finger : less sensitive
  * 5 : Highest level for detect wet finger : more sensitive */
 ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL,
 /* Change threshold for each wet detect level.
  * The valid range is between 10 and 1000. The default is "50 100 150 200 250"  [
Get and set.]
  * 50 : Threshold of lowest level for detect wet finger
  * 250 : Threshold of highest level for detect wet finger */
 ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL_THRESHOLD,
 /* Control rolling area vertically.
  * The valid range is between 0 and 9. The default is 0.  [Get and set.]
  * 0 : minimum position
  * 9 : maximum position */
 ENUM_IBSU_PROPERTY_START_POSITION_OF_ROLLING_AREA,
 /* Enable rolling without lock.
  * The default is FALSE.  [Get and set.] */
 ENUM_IBSU_PROPERTY_START_ROLL_WITHOUT_LOCK,
 /* Enable TOF function.
  * The default is set depending on the devices.  [Get and set.] */
 ENUM_IBSU_PROPERTY_ENABLE_TOF,
/* Enable Encryption for capture images
* The default is FALSE.  [Get and set.] */
 ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION,
/* Reserved for manufacturer strings. [Need a reserve code]*/
ENUM_IBSU_PROPERTY_RESERVED_1 = 200,
ENUM_IBSU_PROPERTY_RESERVED_2,
ENUM_IBSU_PROPERTY_RESERVED_100,
```

/* The previmage processing threshold. [Need a partner or reserve code] The valid range is between 0 and 2, inclusive, with the default of 0 on embedded processor (ARM, Android and Windows Mobile), and with the default of 2 on PC. [Get and set.]

0 : IMAGE_PROCESS_LOW

1 : IMAGE_PROCESS_MEDIUM

2 : IMAGE_PROCESS_HIGH */

ENUM_IBSU_PROPERTY_RESERVED_IMAGE_PROCESS_THRESHOLD = 400,

/* Enable TOF for roll capture

 * The default is FALSE. [Get and set.] */

ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TOF_FOR_ROLL,

/* Change brightness threshold for flat capture

 * The default values are depending on the scanner. [Get and set.] */

ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_FLAT,

/* Change brightness threshold for roll capture

 * The default values are depending on the scanner. [Get and set.] */

ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_ROLL,

/* Change result image to be enhanced

 * The default values are FALSE. [Get and set.] */

ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Note***

  Only specific property values can be set.

1.1.2.11) IBSU_GetProperty

API DLL	int IBSU_GetProperty(const int handle, const IBSU_PropertyId propertyId, LPSTR propertyValue)

- ***Description***

Retrieves a particular device's property value (by handle).

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
propertyId	[in] Property identifier to get value for.
propertyValue	***[out] String returning property value. Memory must be provided by caller.***

- ***IBSU_PropertyId Enumerations (Settable)***

```
/* Product name string (e.g., "Watson").    [Get only.] */
ENUM_IBSU_PROPERTY_PRODUCT_ID,
/* Serial number string.    [Get only.] */
ENUM_IBSU_PROPERTY_SERIAL_NUMBER,
/* Device manufacturer identifier.    [Get only.] */
ENUM_IBSU_PROPERTY_VENDOR_ID,
/* IBIA vendor ID.    [Get only.] */
ENUM_IBSU_PROPERTY_IBIA_VENDOR_ID,
/* IBIA version information.    [Get only.] */
ENUM_IBSU_PROPERTY_IBIA_VERSION,
/* IBIA device ID.    [Get only.] */
ENUM_IBSU_PROPERTY_IBIA_DEVICE_ID,
/* Firmware version string.    [Get only.] */
ENUM_IBSU_PROPERTY_FIRMWARE,
/* Device revision string.    [Get only.] */
ENUM_IBSU_PROPERTY_REVISION,
/* Production date string.    [Get only.] */
ENUM_IBSU_PROPERTY_PRODUCTION_DATE,
/* Last service date string.    [Get only.] */
ENUM_IBSU_PROPERTY_SERVICE_DATE,
/* Image width value.    [Get only.] */
ENUM_IBSU_PROPERTY_IMAGE_WIDTH,
/* Image height value.    [Get only.] */
ENUM_IBSU_PROPERTY_IMAGE_HEIGHT,
/* Time to acquire fingerprint in the auto capture regardless of number of fingers.    The
option IBSU_OPTION_AUTO_CAPTURE must be used.    The valid range is between 2000-
```

and 4000-ms, inclusive, with the default of 4000-ms. [Get and set.] */

ENUM_IBSU_PROPERTY_IGNORE_FINGER_TIME

/* Auto contrast level value. [Get and set.] */

ENUM_IBSU_PROPERTY_RECOMMENDED_LEVEL

/* Polling time for IBSU_BGetImage(). [Get only.] */

ENUM_IBSU_PROPERTY_POLLINGTIME_TO_BGETIMAGE,

/* Enable power save mode (TRUE to enable or FALSE to disable). [Get and set.] */

ENUM_IBSU_PROPERTY_ENABLE_POWER_SAVE_MODE

/* Retry count for communication error. The valid range is between 0 and 120, inclusi ve, with the default of 6. [Get and set.] */

ENUM_IBSU_PROPERTY_RETRY_WRONG_COMMUNICATION

/* The maximum wait time for image capture, in seconds. Must use IBSU_CallbackRe sultImageEx instead of IBSU_CallbackResultImage. If -1, the timeout is infinite. Other wise, the valid range is between 10- and 3600-seconds, inclusive. The default is -1. [Get and set.] */

ENUM_IBSU_PROPERTY_CAPTURE_TIMEOUT

/* Minimum distance of rolled fingerprint, in millimeters. The valid range is between 1 0- and 30-mm, inclusive. The default is 15-mm. [Get and set.] */

ENUM_IBSU_PROPERTY_ROLL_MIN_WIDTH

/* roll mode. The valid range is between 0 ~ 1. The default is 1. [Get and set.]

0 : no use smear

1 : use notice */

ENUM_IBSU_PROPERTY_ROLL_MODE,

/* roll level. The valid range is between 0 ~ 2. The default is 1. [Get and set.]

0 : low level

1 : medium level

2 : high level */

ENUM_IBSU_PROPERTY_ROLL_LEVEL,

/* The area threshold for image capture for flat fingers. The area threshold for beginni ng rolled finger. The valid range is between 0 and 12, inclusive, with the default of 6. [Get and set.] */

ENUM_IBSU_PROPERTY_CAPTURE_AREA_THRESHOLD,

/* Enable decimation mode (TRUE to enable or FALSE to disable). Some of devices (or firmware version) does not support this feature. [Get and set.]*/

ENUM_IBSU_PROPERTY_ENABLE_DECIMATION,

/* Enable capture on release (TRUE to enable or FALSE to disable). The default is F

```
ALSE.   [Get and set.]
TRUE: the result callback will be called when user release the finger from the sensor.
FALSE: the result callback will be called when the quality of finger become good */
ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_ON_RELEASE,
/* The device index. [Get only.] */
ENUM_IBSU_PROPERTY_DEVICE_INDEX,
/* The device ID which has same information with UsbDevice class of Android. [Get o
nly.] */
ENUM_IBSU_PROPERTY_DEVICE_ID,
/* It can be used for dry finger. Some of devices (or firmware version) does not suppo
rt this feature. The default is FALSE. [Get and set.]
TRUE   : Enable dry mode.
FALSE : Disable dry mode */
ENUM_IBSU_PROPERTY_SUPER_DRY_MODE,
/* It is a minimum capture time when the dry mode is enabled with the property ENU
M_IBSU_PROPERTY_SUPER_DRY_MODE. Some of devices (or firmware version) doe
s not support this feature. The valid range is between 600- and 3000-ms, inclusive, wit
h the default of 2000-ms. [Get and set.]*/
ENUM_IBSU_PROPERTY_MIN_CAPTURE_TIME_IN_SUPER_DRY_MODE,
/* Rolled image width value.   [Get only.] */
ENUM_IBSU_PROPERTY_IMAGE_WIDTH,
/* Rolled image height value.   [Get only.] */
ENUM_IBSU_PROPERTY_IMAGE_HEIGHT,
/* Rolled image width value.   [Get only.] */
ENUM_IBSU_PROPERTY_ROLLED_IMAGE_WIDTH,
/* Rolled image height value.   [Get only.] */
ENUM_IBSU_PROPERTY_ROLLED_IMAGE_HEIGHT,
/* Enable the drawing for preview image (TRUE to enable or FALSE to disable).
   The default is TRUE.   [Get and set.] */
ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE,
/* Enable to override roll image (TRUE to enable or FALSE to disable).
   The default is FALSE.   [Get and set.] */
ENUM_IBSU_PROPERTY_ROLL_IMAGE_OVERRIDE,
/* Enable the warning message for invalid area for result image (TRUE to enable or F
ALSE to disable).
   The default is FALSE.   [Get and set.] */
```

```
ENUM_IBSU_PROPERTY_WARNING_MESSAGE_INVALID_AREA,

/* Enable wet detect function.
   The default is FALSE.  [Get and set.] */
ENUM_IBSU_PROPERTY_ENABLE_WET_FINGER_DETECT,

/* Change wet detect level.
  * The valid range is between 1 and 5. The default is 3.  [Get and set.]
  * 1 : Lowest level for detect wet finger : less sensitive
  * 5 : Highest level for detect wet finger : more sensitive */
 ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL,

 /* Change threshold for each wet detect level.
  * The valid range is between 10 and 1000. The default is "50 100 150 200 250"  [
Get and set.]
  * 50 : Threshold of lowest level for detect wet finger
  * 250 : Threshold of highest level for detect wet finger */
 ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL_THRESHOLD,

 /* Control rolling area vertically.
  * The valid range is between 0 and 9. The default is 0.  [Get and set.]
  * 0 : minimum position
  * 9 : maximum position */
 ENUM_IBSU_PROPERTY_START_POSITION_OF_ROLLING_AREA,

 /* Enable rolling without lock.
  * The default is FALSE.  [Get and set.] */
 ENUM_IBSU_PROPERTY_START_ROLL_WITHOUT_LOCK,

 /* Enable TOF function.
  * The default is set depending on the devices.  [Get and set.] */
 ENUM_IBSU_PROPERTY_ENABLE_TOF,

/* Enable Encryption for capture images
* The default is FALSE.   [Get and set.] */
 ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION,

/* Reserved for manufacturer strings. [Need a reserve code]*/
ENUM_IBSU_PROPERTY_RESERVED_1 = 200,
ENUM_IBSU_PROPERTY_RESERVED_2,
ENUM_IBSU_PROPERTY_RESERVED_100,

/* The previmage processing threshold. [Need a partner or reserve code]  The valid ra
nge is between 0 and 2, inclusive, with the default of 0 on embedded processor (AR
M, Android and Windows Mobile), and with the default of 2 on PC. [Get and set.]
0  : IMAGE_PROCESS_LOW
1  : IMAGE_PROCESS_MEDIUM
2  : IMAGE_PROCESS_HIGH */
ENUM_IBSU_PROPERTY_RESERVED_IMAGE_PROCESS_THRESHOLD = 400,
/* Enable TOF for roll capture
     * The default is FALSE.  [Get and set.] */
 ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TOF_FOR_ROLL,
 /* Change brightness threshold for flat capture
     * The default values are depending on the scanner.  [Get and set.] */
 ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_
FLAT,
 /* Change brightness threshold for roll capture
     * The default values are depending on the scanner.  [Get and set.] */
 ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_
ROLL,
 /* Change result image to be enhanced
     * The default values are FALSE.  [Get and set.] */
 ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE
```

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
// Get Image size to be supported in device
nRc = IBSU_GetProperty(pDialog->m_DeviceHandle,
                    ENUM_IBSU_PROPERTY_IMAGE_WIDTH, &cValue[0] );
Width = atoi(cValue);
nRc = IBSU_GetProperty(pDialog->m_DeviceHandle,
                    ENUM_IBSU_PROPERTY_IMAGE_HEIGHT, &cValue[0] );
Height = atoi(cValue);
```

1.1.2.12) IBSU_AsyncOpenDevice

- ***Prototype***

API DLL	int IBSU_AsyncOpenDevice(const int deviceIndex,)

- ***Description***

Asynchronous Initialize device, given a particular by device index.

- ***Parameter***

Parameter	Description
index	[in] Zero-based device index for device to init.

- ***Return***

Return Value	Description
0	Device is ready to be used.
< 0	The error code as defined in IBScanUltimateApi_err.h
> 0	Indicates that the device was already initialized and can be used

- ***Note***

Any initialized device must be released before shutting down the application call by IBSU_CloseDevice() or IBSU_CloseAlldevice().

1.1.2.13) IBSU_OpenDeviceEx

- ***Prototype***

API DLL	int WINAPI IBSU_OpenDeviceEx (const int deviceIndex, LPCSTR uniformityMaskPath, const BOOL asyncOpen, int *pHandle)

- ***Description***

Extension initialize device(fast mode), given a particular by device index.

- ***Parameter***

Parameter	Description
deviceIndex	[in] Zero-based device index for device to init
uniformityMaskPath	[in] Uniformity mask path in your computer If the file does not exist or different in path, the DLL makes a new

	file in path.
asyncOpen	[in] async open device(TRUE) or sync open device(FALSE)
*pHandle	[out] Function returns device handle to be used for subsequent function calls. Memory must be provided by caller

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Note***

  Any initialized device must be released before closing the host application! (call IBSU_CloseDevice() or IBSU_CloseAlldevice())

1.1.2.14) IBSU_EnableTraceLog

- ***Prototype***

API DLL	int IBSU_EnableTraceLog(BOOL on)

- ***Description***

  Enables or disables trace log. The trace log is enabled by default

- ***Parameter***

Parameter	Description
on	[in] TRUE to enable trace log; FALSE to disable it

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.2.15) IBSU_UnloadLibrary

- ***Prototype***

API DLL	int IBSU_FreeLibrary()

- ***Description***

  The library is unmapped from the address space explicitly, and the library is no longer valid

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

2.1.3. Image Acquisition Related Interface Functions

1.1.3.1) IBSU_IsCaptureAvailable

- ***Prototype***

API DLL	int IBSU_IsCaptureAvailable (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, BOOL *pIsAvailable)

- ***Description***

  Check if a requested capture mode is supported by the device.

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
imageType	[in] Image type to verify.
imageResolution	[in] Requested capture resolution.
*pIsAvailable	***[out] Returns TRUE if mode is available. Memory must be provided by caller.***

- ***IBSU_ImageType Enumerations***

```
/* Unspecified type. */
ENUM_IBSU_TYPE_NONE,
/* One-finger rolled fingerprint. */
ENUM_IBSU_ROLL_SINGLE_FINGER,
/* One-finger flat fingerprint. */
ENUM_IBSU_FLAT_SINGLE_FINGER,
/* Two-finger flat fingerprint. */
ENUM_IBSU_FLAT_TWO_FINGERS,
/* Four-finger flat fingerprint. */
ENUM_IBSU_FLAT_FOUR_FINGERS
```

- ***IBSU ImageResolution Enumerations***

```
/* 500 pixels/inch. */
ENUM_IBSU_IMAGE_RESOLUTION_500 = 500
```

- ***Return***

Return Value	Description
0	Function completed successfully.

< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
BOOL modeAvailable;
IBSU_IsCaptureAvailable( m_DeviceHandle, imageType,
                    ENUM_IBSU_IMAGE_RESOLUTION_500, &modeAvailable );
if( !modeAvailable )
{
          _SetStatusBarText( _T( "Capture mode %d not available" ),
                                        imageType );
          OnUpdateScreenElements();
          return 0L;
}
```

1.1.3.2) IBSU_BeginCaptureImage

- ***Prototype***

API DLL	int IBSU_BeginCaptureImage (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, const DWORD captureOptions);

- ***Description***

Starts image acquisition for a particular device (by handle).

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
imageType	[in] Image type to capture.
imageResolution	[in] Requested capture resolution.
captureOptions	[in] Bit coded capture options to use

- ***IBSU_ImageType Enumerations***

```
/* Unspecified type. */
ENUM_IBSU_TYPE_NONE,
/* One-finger rolled fingerprint. */
ENUM_IBSU_ROLL_SINGLE_FINGER,
/* One-finger flat fingerprint. */
```

```
ENUM_IBSU_FLAT_SINGLE_FINGER,
/* Two-finger flat fingerprint. */
ENUM_IBSU_FLAT_TWO_FINGERS,
/* Four-finger flat fingerprint. */
ENUM_IBSU_FLAT_FOUR_FINGERS
```

- ***IBSU ImageResolution Enumerations***

```
/* 500 pixels/inch. */
ENUM_IBSU_IMAGE_RESOLUTION_500 = 500
```

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *IBSU_ERR_CAPTURE_STILL_RUNNING -> an acquisition is currently pending and needs to be completed first* *IBSU_ERR_INVALID_PARAM_VALUE -> parameter numberOfObjects needs to be in range 1..2* *IBSU_ERR_CHANNEL_INVALID_MODE -> acquisition mode needs to be set as a prerequisite*

- ***Example***

```
nRc = IBSU_IsTouchedFinger(pDialog->m_DeviceHandle, &touchInStatus);
if( nRc == IBSU_STATUS_OK )
{
        if( touchInStatus == 1 )
        {
                nRc = IBSU_BeginCaptureImage(
                        pDialog->m_DeviceHandle, imageType,
                        ENUM_IBSU_IMAGE_RESOLUTION_500, captureOptions);
                if( nRc >= IBSU_STATUS_OK )
                {
                        // Display instructions
                        CString strMessage;
                        pDialog->_SetStatusBarText( _T( "Succeed to
                                execute capture start by touch sensor" ) );
                }
                else
                        pDialog->_SetStatusBarText( _T( "Failed to
                              execute capture start by touch sensor" ) );
                              break;
                    }
          }
          Sleep(10);
```

- ***Note***

  Once image acquisition is completed, image streaming will continue in the background (to minimize delays when restarting acquisition). In order to stop communication traffic on the PC bus system, streaming can be stopped by setting the capture mode to ENUM_IBSU_TYPE_NONE.

1.1.3.3) IBSU_CancelCaptureImage

- ***Prototype***

API DLL	int IBSU_CancelCaptureImage (const int handle)

- ***Description***

  Abort image acquisition on a device that is currently scanning.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *IBSU_ERR_CAPTURE_NOT_RUNNING -> no active acquisition to be aborted*

- ***Example***

```
//      Stop button pressed.
void CIBScanUltimate_ImplementationDlg::OnBtnStop()
{
          if( m_DeviceHandle == -1 )
          {
                    //      ASSERT( FALSE );
                    return;
          }

          IBSU_CancelCaptureImage( m_DeviceHandle );
          m_CurrentStep = -1;

          _SetStatusBarText( _T( "Sequence aborted" ) );
          OnUpdateScreenElements();
  }
```

1.1.3.4) IBSU_ IsCaptureActive

- ***Prototype***

API DLL	int IBSU_IsCaptureActive(const int handle, BOOL *pIsActive)

- ***Description***

  Check if a particular device is actively scanning for image acquisition.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
*pIsActive	***[out] Returns TRUE if acquisition is in progress (preview or result image acquisition). Memory must be provided by caller.***

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
BOOL IsActive;
int nRc;
nRc = IBSU_IsCaptureActive(m_DeviceHandle, &IsActive);
if( nRc == IBSU_STATUS_OK && IsActive )
{
        IBSU_TakeResultImageManually(m_DeviceHandle);
}
else
{
          // device already initialized
          // -> directly begin acquisition sequence
          PostMessage( WM_USER_CAPTURE_READY );
}
```

1.1.3.5) IBSU_TakeResultImageManually

- ***Prototype***

API DLL	int IBSU_TakeResultImageManually (const int handle)

- ***Description***

  Start image acquisition for a particular device (by handle) with image gain manually set.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
BOOL IsActive;
int nRc;
nRc = IBSU_IsCaptureActive(m_DeviceHandle, &IsActive);
if( nRc == IBSU_STATUS_OK && IsActive )
{
          IBSU_TakeResultImageManually(m_DeviceHandle);
}
else
{
          // device already initialized
          // -> directly begin acquisition sequence
```

```
                PostMessage( WM_USER_CAPTURE_READY );
        }
```

1.1.3.6) IBSU_GetContrast

- ***Prototype***

API DLL	int IBSU_GetContrast (const int handle, int *pContrastValue)

- ***Description***

  Get the contrast value for a particular scanner.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
*pContrastValue	***[out] Contrast value (range: 0 <= value <= IBSU_MAX_CONTRAST_VALUE). Memory must be provided by caller.***

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *IBSU_ERR_CHANNEL_NOT_ACTIVE -> acquisition channel needs to be selected as a prerequisite.*

- ***Example***

```
// Button click event to check the current contrast level.
void CIBScanUltimate_ImplementationDlg::OnBnClickedBtnGetContrast()
{
        int contrastValue;
        int nRc = IBSU_GetContrast( m_DeviceHandle, &contrastValue );
        if( nRc >= IBSU_STATUS_OK )
        {
                m_Contrast = contrastValue;
        }
        _SetStatusBarText( _T("-- GetContrast() --\tReturn value = %d"),
                                                nRc );
}
```

1.1.3.7) IBSU_SetContrast

- ***Prototype***

API DLL	int IBSU_SetContrast (const int handle, const int contrastValue)

- ***Description***

  Set the contrast value for a particular scanner.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
contrastValue	[in] Contrast value (range: 0 <= value <= IBSU_MAX_CONTRAST_VALUE)

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *IBSU_ERR_CHANNEL_NOT_ACTIVE -> acquisition channel needs to be selected as a prerequisite*

- ***Example***

```
// Button click event to set the current contrast level.
void CIBScanUltimate_ImplementationDlg::OnBnClickedBtnSetContrast()
{
        int nRc = IBSU_SetContrast(m_DeviceHandle, m_Contrast);
        _SetStatusBarText( _T("-- SetContrast() --\tReturn value = %d"),
                                          nRc );
}
```

1.1.3.8) IBSU_SetLEOperationMode

- ***Prototype***

API DLL	int IBSU_SetLEOperationMode(const int handle, const IBSU_LEOperationMode leOperationMode)

- ***Description***

Sets the touch operation mode (On, Off, or Auto) for a particular scanner.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
leOperationMode	[in] LE film operation mode

- ***IBSU LEOperationMode Enumerations***

ENUM_IBSU_LE_OPERATION_AUTO,

ENUM_IBSU_LE_OPERATION_ON,

ENUM_IBSU_LE_OPERATION_OFF

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
// Radio button click event to set auto touch mode.
void CIBScanUltimate_ImplementationDlg::OnBnClickedRadioTouchAuto()
{
            int nRc = IBSU_SetLEOperationMode(m_DeviceHandle,
            ENUM_IBSU_LE_OPERATION_AUTO);
            _SetStatusBarText( _T("-- SetLEOperationMode(AUTO) --\tReturn
                                                  value = %d"), nRc );
}
```

1.1.3.9) IBSU_GetLEOperationMode

- ***Prototype***

API DLL	int IBSU_GetLEOperationMode(const int handle, IBSU_LEOperationMode *leOperationMode)

- ***Description***

Gets the touch operation mode (On, Off, or Auto) for a particular scanner.

- ***Parameters***

Parameter	Description

handle	[in] Device handle obtained by IBSU_OpenDevice()
*leOperationMode	[out] LE film operation mode (Memory must be provided by caller).

- ***IBSU LEOperationMode Enumerations***

  ENUM_IBSU_LE_OPERATION_AUTO,

  ENUM_IBSU_LE_OPERATION_ON,

  ENUM_IBSU_LE_OPERATION_OFF

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
// Button click event to check the current operation mode.
void CIBScanUltimate_ImplementationDlg::OnBnClickedBtnGetLeMode()
{
        IBSU_LEOperationMode leOperationMode;
        CString str;
        int nRc = IBSU_GetLEOperationMode(m_DeviceHandle,
                                                    &leOperationMode);
        _SetStatusBarText( _T("-- GetLEOperationMode --\tReturn
                                          value = %d"), nRc );
        if( nRc == IBSU_STATUS_OK )
        {
                if( leOperationMode == ENUM_IBSU_LE_OPERATION_AUTO )
                        str = "AUTO";
                else if( leOperationMode == ENUM_IBSU_LE_OPERATION_ON )
                        str = "ON";
                else if( leOperationMode == ENUM_IBSU_LE_OPERATION_OFF )
                        str = "OFF";
                else
                        str = "Unknown";

                GetDlgItem(IDC_EDIT_LE_MODE)->SetWindowText(str);
        }
}
```

1.1.3.10) IBSU_IsTouchedFinger

- ***Prototype***

API DLL	int IBSU_IsTouchedFinger (const inthandle, int *touchInValue)

- ***Description***

  Queries a particular scanner to determine if a finger is currently detected.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
IeOperationMode	[out] touchValue value (0 : touch off, 1 : touch on). Memory must be provided by caller

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

- ***Example***

```
// Button click event to currently detect the finger.
void CIBScanUltimate_ImplementationDlg::OnBnClickedBtnGetTouchIn()
{
        int touchInStatus;
        CString str;
        int nRc = IBSU_IsTouchedFinger(m_DeviceHandle, &touchInStatus);
        _SetStatusBarText( _T("-- IsTouchedFinger() --\tReturn
                                        value = %d"), nRc );

        if( nRc == IBSU_STATUS_OK )
        {
                if( touchInStatus == 0 )
                        str = "Not detected";
                else if( touchInStatus == 1 )
                        str = "Finger detected";
                else
                        str = "unknown";
```

```
                    GetDlgItem(IDC_TXT_TOUCH_STATUS)->SetWindowText(str);
          }
          else
          {
                    GetDlgItem(IDC_TXT_TOUCH_STATUS)->SetWindowText("");
          }
}
```

1.1.3.11) IBSU_GetOperableLEDs

- ***Prototype***

API DLL	int IBSU_GetOperableLEDs (const int handle, IBSU_LedType *pLedType, int *pLedCount, DWORD *pOperableLEDs)

- ***Description***

Get operable status LED's.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
*pLedType	[out] Type of LED's. Memory must be provided by caller.
*pLedCount	[out] Number of LED's. Memory must be provided by caller.
*pOperableLEDs	[out] Bit pattern of operable LED's. Memory must be provided by caller.

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.12) IBSU_GetLEDs

- ***Prototype***

API DLL	int IBSU_IsTouchedFinger (const int handle, DWORD *pActiveLEDs)

- ***Description***

  Get active status LED's for device.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
*pActiveLEDs	[out] get active LEDs. Memory must be provided by caller.

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.13) IBSU_SetLEDs

- ***Prototype***

API DLL	int IBSU_SetLEDs (const int handle, const DWORD activeLEDs)

- ***Description***

  Set active status LED's on device.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
ActiveLEDs	[in] set active LEDs.

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.14) IBSU_GenerateZoomOutImage

- ***Prototype***

API DLL	Int WINAPI IBSU_GnerateZoomOutImage (const

	IBSU_ImageData inImage, BYTE *outImage, const int outWidth, const int outHeight, const BYTE bkColor)

- ***Description***

  Make a smaller image of a fingerprint scan.

- ***Parameters***

Parameter	Description
inImage	[in] Original image
*outImage	[out] Pointer to zoom-out image data buffer (Memory must be provided by caller)
outWidth	[in] width for zoom-out image
outHeight	[in] height for zoom-out image
bkColor	[in] Background color for remain area from zoom-out image

- ***IBSU_ ImageData Structure Definition***

```
typedef struct tagIBSU_ImageData
{
    /* Pointer to image buffer.   If this structure is supplied by a callback function, this pointer
    must not be retained; the data should be copied to an application buffer for any
    processing after the callback returns. */
    void                *Buffer;
    /* Image horizontal size (in pixels). */
    DWORD               Width;
    /* Image vertical size (in pixels). */
    DWORD               Height;
    /* Horizontal image resolution (in pixels/inch). */
    double              ResolutionX;
    /* Vertical image resolution (in pixels/inch). */
    double              ResolutionY;
    /* Image acquisition time, excluding processing time (in seconds). */
    double              FrameTime;
    /* Image line pitch (in bytes).   A positive value indicates top-down line order; a negative
    value indicates bottom-up line order. */
    int                 Pitch;
    /* Number of bits per pixel. */
    BYTE                BitsPerPixel;
    /* Image color format. */
    IBSU_ImageFormat   Format;
    /* Marks image as the final processed result from the capture.   If this is FALSE, the
    image is a preview image or a preliminary result. */
    BOOL                    IsFinal;
    /* Threshold of image processing. */
    DWORD                   ProcessThres;
  }
```

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.15) IBSU_SaveBitmapMem

- ***Prototype***

API DLL	int WINAPI IBSU_SaveBitmapMem (const BYTE *inImage, const DWORD inWidth, const DWORD inHeight, const int inPitch, const double inResX, const double inResY, BYTE *outBitmapBuffer, const IBSU_ImageFormat outImageFormat, const DWORD outWidth, const DWORD outHeight, const BYTE bkColor);

- ***Description***

Save fingerprint image in bitmap format.

- ***Parameters***

Parameter	Description
*inImage	[in] Point to image data (Gray scale image)
inWidth	[in] Image width (in pixels)
inHeight	[in] Image height (in pixels)

inPitch	[in] Image line pitch (in bytes) (Positive value indicate top down line order, Negative value mean bottom up line order)
inResX	[in] Image horizontal resolution (in PPI)
inResY	[in] Image vertical resolution (in PPI)
*outBitmapBuffer	[out] Pointer to output image data buffer
outImageFormat	[in] Set Image color format for output image
outWidth	[in] Width for zoom-out image
outHeight	[in] height for zoom-out image
bkColor	[in] Background color for remain area from zoom-out image

- ***IBSU_ImageFormat Enumerations***

```
IBSU_IMG_FORMAT_GRAY,     /* Gray-scale image. */
IBSU_IMG_FORMAT_RGB24,    /* 24-bit color image. */
IBSU_IMG_FORMAT_RGB32,    /* True-color RGB image. */
IBSU_IMG_FORMAT_UNKNOWN   /* Unknown format. */
```

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.16) IBSU_SaveBitmapImage

- ***Prototype***

API DLL	Int WINAPI IBSU_SaveBitmapImage (LPCSTR filepath, const BYTE *imgBuffer, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)

- ***Description***

Save fingerprint image in bitmap format.

- ***Parameters***

Parameter	Description
filePath	[in] File path to save bitmap
*imgBuffer	[in] Point to raw image data (background color is black)
Width	[in] Image width

Height	[in] Image height
Pitch	[in] Image line pitch (Positive value indicate top down line order, Negative value mean bottom up line order)
resX	[in] Image horizontal resolution (in PPI)
resY	[in]Image vertical resolution (in PPI)

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.17) IBSU_BGetImage

- ***Prototype***

API DLL	Int WINAPI IBSU_GetImage (const int handle, IBSU_ImageData *pImage, IBSU_ImageType *pImageType, IBSU_ImageData *pSplitImageArray, Int *pSplitImageArrayCount, IBSU_FingerCountState *pFingerCountState, IBSU_FingerQualityState *pQualityArray, Int *pQualityArrayCount);

- ***Description***

Get image with non-blocking function (with IBSU_AsyncOpenDevice()).

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*pImage	[out] Image data of preview image or result image
*pImageType	[out] Image type
*pSplitImageArray	[out] Finger array to be split from result image (two-fingers, four-fingers)
*pSplitImageArray	[out] Array count to be split from result image (two-fingers, four-

Count	fingers)
*pFingerCountState	[out] Finger count state
*pQualityArray	[out] Finger quality state
*pQualityArrayCount	[out] Finger quality count

- ***IBSU_ ImageData Structure Definition***

```
typedef struct tagIBSU_ImageData
{
    /* Pointer to image buffer.   If this structure is supplied by a callback function, this pointer
    must not be retained; the data should be copied to an application buffer for any
    processing after the callback returns. */
    void                  *Buffer;
    /* Image horizontal size (in pixels). */
    DWORD                      Width;
    /* Image vertical size (in pixels). */
    DWORD                      Height;
    /* Horizontal image resolution (in pixels/inch). */
    double                   ResolutionX;
    /* Vertical image resolution (in pixels/inch). */
    double                   ResolutionY;
    /* Image acquisition time, excluding processing time (in seconds). */
    double                   FrameTime;
    /* Image line pitch (in bytes).   A positive value indicates top-down line order; a negative
    value indicates bottom-up line order. */
    int                      Pitch;
    /* Number of bits per pixel. */
    BYTE                       BitsPerPixel;
    /* Image color format. */
    IBSU_ImageFormat   Format;
    /* Marks image as the final processed result from the capture.   If this is FALSE, the
    image is a preview image or a preliminary result. */
    BOOL                       IsFinal;
    /* Threshold of image processing. */
    DWORD                      ProcessThres;
 }
```

- ***IBSU_ImageType Enumerations***

  /* Unspecified type. */

  ENUM_IBSU_TYPE_NONE,

  /* One-finger rolled fingerprint. */

  ENUM_IBSU_ROLL_SINGLE_FINGER,

  /* One-finger flat fingerprint. */

  ENUM_IBSU_FLAT_SINGLE_FINGER,

  /* Two-finger flat fingerprint. */

  ENUM_IBSU_FLAT_TWO_FINGERS,

  /* Four-finger flat fingerprint. */

  ENUM_IBSU_FLAT_FOUR_FINGERS

- ***IBSU_ FingerCountState Enumerations***

  ENUM_IBSU_FINGER_COUNT_OK,

  ENUM_IBSU_TOO_MANY_FINGERS,

  ENUM_IBSU_TOO_FEW_FINGERS,

  ENUM_IBSU_NON_FINGER

- ***IBSU_ FingerQualityState Enumerations***

  ENUM_IBSU_FINGER_NOT_PRESENT,

  ENUM_IBSU_QUALITY_GOOD,

  ENUM_IBSU_QUALITY_FAIR,

  ENUM_IBSU_QUALITY_POOR,

  /* Finger position is not valid on top side. */

  ENUM_IBSU_QUALITY_INVALID_AREA_TOP,

  /* Finger position is not valid on left side. */

  ENUM_IBSU_QUALITY_INVALID_AREA_LEFT,

  /* Finger position is not valid on right side. */

  ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT,

  /* Finger position is not valid on bottom side. */

  ENUM_IBSU_QUALITY_INVALID_AREA_BOTTOM

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.18) IBSU_BGetImageEx

- ***Prototype***

API DLL	Int WINAPI IBSU_GetImageEx(const int handle, (const int handle, int *pImageStatus, IBSU_ImageData *pImage, IBSU_ImageType *pImageType, int *pDetectedFingerCount, IBSU_ImageData *pSegmentImageArray, IBSU_SegmentPosition *pSegmentPositionArray, int *pSegmentImageArrayCount, IBSU_FingerCountState *pFingerCountState, IBSU_FingerQualityState *pQualityArray, int *pQualityArrayCount)

- ***Description***

Acquire an image from a device, blocking for result. The segment image array will only be populated if the image is a result image, i.e., if the IsFinal member of pImage is set to TRUE.

- ***Parameters***

Parameter	Description
handle	Device handle
*pImageStatus	Pointer to variable that will receive status from result image acquisition. See error codes in *IBScanUltimateApi_err.h*.
*pImage	Pointer to structure that will receive data of preview or result image. The buffer in this structure points to an internal image buffer; the data should be copied to an application buffer if desired for future processing.
*pImageType	Pointer to variable that will receive image type.
*pDetectedFingerCount	Pointer to variable that will receive detected finger count.
*pSegmentImageArray	Pointer to array of four structures that will receive individual finger image segments from result image. The buffers in these structures point to internal

	image buffers; the data should be copied to application buffers if desired for future processing.
*pSegmentPositionArray	Pointer to array of four structures that will receive data for individual fingers split from result image.
*pSegmentImageArrayCount	Pointer to variable that will receive number of finger images split from result image.
*pFingerCountState	Pointer to variable that will receive finger count state.
*pQualityArray	Pointer to array of four variables that will receive quality states for finger images.
*pQualityArrayCount	Pointer to variable that will receive number of finger qualities.

- ***IBSU_ ImageData Structure Definition***

```
typedef struct tagIBSU_ImageData
{
    /* Pointer to image buffer.   If this structure is supplied by a callback function, this pointer
    must not be retained; the data should be copied to an application buffer for any
    processing after the callback returns. */
    void                 *Buffer;
    /* Image horizontal size (in pixels). */
    DWORD                Width;
    /* Image vertical size (in pixels). */
    DWORD                Height;
    /* Horizontal image resolution (in pixels/inch). */
    double               ResolutionX;
    /* Vertical image resolution (in pixels/inch). */
    double               ResolutionY;
    /* Image acquisition time, excluding processing time (in seconds). */
    double               FrameTime;
    /* Image line pitch (in bytes).   A positive value indicates top-down line order; a negative
    value indicates bottom-up line order. */
    int                  Pitch;
    /* Number of bits per pixel. */
    BYTE                 BitsPerPixel;
    /* Image color format. */
    IBSU_ImageFormat   Format;
    /* Marks image as the final processed result from the capture.   If this is FALSE, the
    image is a preview image or a preliminary result. */
    BOOL                    IsFinal;
    /* Threshold of image processing. */
    DWORD                   ProcessThres;
  }
```

- ***IBSU_ImageType Enumerations***

```
/* Unspecified type. */
ENUM_IBSU_TYPE_NONE,
/* One-finger rolled fingerprint. */
ENUM_IBSU_ROLL_SINGLE_FINGER,
/* One-finger flat fingerprint. */
ENUM_IBSU_FLAT_SINGLE_FINGER,
/* Two-finger flat fingerprint. */
ENUM_IBSU_FLAT_TWO_FINGERS,
/* Four-finger flat fingerprint. */
ENUM_IBSU_FLAT_FOUR_FINGERS
```

- ***IBSU_ SegmentPosition Structure Definition***

```
typedef struct tagIBSU_SegmentPosition
{
    short x1;        /* X coordinate of starting point of the finger segment. */
    short y1;        /* Y coordinate of starting point of the finger segment. */
    short x2;        /* X coordinate of 1st corner of the finger segment. */
    short y2;        /* Y coordinate of 1st corner of the finger segment. */
    short x3;        /* X coordinate of 2nd corner of the finger segment. */
    short y3;        /* Y coordinate of 2nd corner of the finger segment. */
    short x4;        /* X coordinate of 3rd corner of the finger segment. */
    short y4;        /* Y coordinate of 3rd corner of the finger segment. */
}
```

- ***IBSU_ FingerCountState Enumerations***

```
ENUM_IBSU_FINGER_COUNT_OK,
ENUM_IBSU_TOO_MANY_FINGERS,
ENUM_IBSU_TOO_FEW_FINGERS,
ENUM_IBSU_NON_FINGER
```

- ***IBSU_ FingerQualityState Enumerations***

```
ENUM_IBSU_FINGER_NOT_PRESENT,
ENUM_IBSU_QUALITY_GOOD,
ENUM_IBSU_QUALITY_FAIR,
ENUM_IBSU_QUALITY_POOR,
/* Finger position is not valid on top side. */
ENUM_IBSU_QUALITY_INVALID_AREA_TOP,
/* Finger position is not valid on left side. */
ENUM_IBSU_QUALITY_INVALID_AREA_LEFT,
/* Finger position is not valid on right side. */
ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT
```

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.19) IBSU_BGetInitProgress

- ***Prototype***

API DLL	Int WINAPI IBSU_GetInitProgress (const int deviceIndex, BOOL *pIsComplete, int *pHandle, int *pProgressValue);

- ***Description***

Get initialize status with non-blocking function (with IBSU_AsyncOpenDevice()).

- ***Parameters***

Parameter	Description
deviceIndex	[in] Device index
*pIsComplete	[out] Is that complete the initialize device
*pHandle	[out] Device handle
*pProgressValue	[out] progress value of initialize device

- ***Returns***

Return Value	Description
0	Function completed successfully.

< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.20) IBSU_BGetClearPlatenAtCapture

- ***Prototype***

API DLL	Int WINAPI IBSU_GetInitProgress (const int handle, IBSU_PlatenState *pPlatenState,);

- ***Description***

  Check there is fingers when start capture image with non-blocking function (with IBSU_AsyncOpenDevice()).

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*pPlatenState	[out] Platen status

- ***IBSU_ PlatenState Enumerations***
  ENUM_IBSU_PLATEN_CLEARD,
  ENUM_IBSU_PLATEN_HAS_FINGERS

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *

1.1.3.21) IBSU_BGetRollingInfo

- ***Prototype***

API DLL	Int WINAPI IBSU_BGetRollingInfo (const int handle, IBSU_RollingState *pRollingState,

	Int *pRollingLineX);

- ***Description***

Rolling information for user drawing.

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*pRollingState	[out] Rolling state
*pRollingLineX	[our] x-coordinate of Rolling line for drawing

- ***IBSU_ RollingState Enumerations***

ENUM_IBSU_ROLLING_NOT_PRESENT,

ENUM_IBSU_ROLLING_TAKE_ACQUISITION,

ENUM_IBSU_ROLLING_COMPLETE_ACQUISITION,

ENUM_IBSU_ROLLING_RESULT_IMAGE

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *

1.1.3.22) IBSU_GetIBSM_ResultImageInfo

- ***Prototype***

API DLL	Int WINAPI IBSU_GetIBSM_ResultImageInfo(const int handle, IBSM_FingerPosition fingerPosition, IBSM_ImageData *pResultImage, IBSM_ImageData *pSplitResultImageCount);

- ***Description***

Result image is made into IBSM_ImageData struct

- ***Parameters***

Parameter	Description
handle	[in] Device handle

fingerPosition	[in] Finger position
*pResultImage	[out] Result image
*pSplitResultImag e	[out] Split image from Result image
*pSplitResultImag e	[out] Split image count

- ***IBSM_ FingerPosition Enumerations***

  IBSM_FINGER_POSITION_UNKNOWN=0,

  IBSM_FINGER_POSITION_RIGHT_THUMB,

  IBSM_FINGER_POSITION_RIGHT_INDEX_FINGER,

  IBSM_FINGER_POSITION_RIGHT_MIDDLE_FINGER,

  IBSM_FINGER_POSITION_RIGHT_RING_FINGER,

  IBSM_FINGER_POSITION_RIGHT_LITTLE_FINGER,

  IBSM_FINGER_POSITION_LEFT_THUMB,

  IBSM_FINGER_POSITION_LEFT_INDEX_FINGER,

  IBSM_FINGER_POSITION_LEFT_MIDDLE_FINGER,

  IBSM_FINGER_POSITION_LEFT_RING_FINGER,

  IBSM_FINGER_POSITION_LEFT_LITTLE_FINGER,

  IBSM_FINGER_POSITION_PLAIN_RIGHT_FOUR_FINGERS=13,

  IBSM_FINGER_POSITION_PLAIN_LEFT_FOUR_FINGERS,

  IBSM_FINGER_POSITION_PLAIN_THUMBS,

  IBSM_FINGER_POSITION_UNKNOWN_PALM=20,

  IBSM_FINGER_POSITION_RIGHT_FULL_PALM,

  IBSM_FINGER_POSITION_RIGHT_WRITERS_PALM,

  IBSM_FINGER_POSITION_LEFT_FULL_PALM,

  IBSM_FINGER_POSITION_LEFT_WRITERS_PALM,

  IBSM_FINGER_POSITION_RIGHT_LOWER_PALM,

  IBSM_FINGER_POSITION_RIGHT_UPPER_PALM,

  IBSM_FINGER_POSITION_LEFT_LOWER_PALM,

  IBSM_FINGER_POSITION_LEFT_UPPER_PALM,

  IBSM_FINGER_POSITION_RIGHT_OTHER,

  IBSM_FINGER_POSITION_LEFT_OTHER,

  IBSM_FINGER_POSITION_RIGHT_INTERDIGITAL,

  IBSM_FINGER_POSITION_RIGHT_THENAR,

  IBSM_FINGER_POSITION_RIGHT_HYPOTHENAR,

  IBSM_FINGER_POSITION_LEFT_INTERDIGITAL,

```
IBSM_FINGER_POSITION_LEFT_THENAR,
IBSM_FINGER_POSITION_LEFT_HYPOTHENAR,
IBSM_FINGER_POSITION_RIGHT_INDEX_AND_MIDDLE=40,
IBSM_FINGER_POSITION_RIGHT_MIDDLE_AND_RING,
IBSM_FINGER_POSITION_RIGHT_RING_AND_LITTLE,
IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE,
IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_RING,
IBSM_FINGER_POSITION_LEFT_RING_AND_LITTLE,
IBSM_FINGER_POSITION_RIGHT_INDEX_AND_LEFT_INDEX,
IBSM_FINGER_POSITION_RIGHT_INDEX_AND_MIDDLE_AND_RING,
IBSM_FINGER_POSITION_RIGHT_MIDDLE_AND_RING_AND_LITTLE,
IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE_AND_RING,
IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_RING_AND_LITTLE
```

- ***IBSM_ ImageData Structure Definition***

```
typedef struct tagIBSM_ImageData
{
  IBSM_ImageFormat          ImageFormat;
  IBSM_ImpressionType       ImpressionType;
  IBSM_FingerPosition       FingerPosition;
  IBSM_CaptureDeviceTechID CaptureDeviceTechID;
  unsigned short            CaptureDeviceVendorID;
  unsigned short            CaptureDeviceTypeID;
  unsigned short            ScanSamplingX;
  unsigned short            ScanSamplingY;
  unsigned short            ImageSamplingX;
  unsigned short            ImageSamplingY;
  unsigned short            ImageSizeX;
  unsigned short            ImageSizeY;
  unsigned char             ScaleUnit;
  unsigned char             BitDepth;
  unsigned int              ImageDataLength;
  void                      *ImageData;
}
```

- ***Returns***

Return Value	Description
0	Function completed successfully.

< 0	The error code as defined in IBScanUltimateApi_err.h [*]

1.1.3.23) IBSU_GetNFIQScore

- ***Prototype***

API DLL	Int WINAPI IBSU_GetNFIQScore(const int handle, const BYTE *imgBuffer, const DWORD width, const DWORD height, const BYTE bitsPerPixel, int *pScore);

- ***Description***

  Return NFIQ score

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*imgBuffer	[in] Point to image data
width	[in] Image width
height	[in] Image height
bitsPerPixel	[in] Number of Bits per pixel
*pScore	[out] NFIQ score

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h [†]

1.1.3.24) IBSU_GenerateZoomOutImageEx

- ***Prototype***

API DLL	Int WINAPI IBSU_GenerateZoomOutImageEx(const BYTE *pInImage, const int inWidth, const int inHeight, BYTE *outImage, const int outWidth, const int outHeight, const BYTE

	bkColor)

- ***Description***

  Make a smaller image of a fingerprint scan.

- ***Parameters***

Parameter	Description
*pInImage	[in] Original image
inWidth	[in] Width of original image
inHeight	[in] Height of original image
*outImage	[out] Pointer to zoom-out image data buffer memory must be provided by caller
outWidth	[in] Width for zoom-out image
outHeight	[in] Height for zoom-out image
bkColor	[in] Background color for remain area from zoom-out image

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *

1.1.3.25) IBSU_WSQEncodeMem

- ***Prototype***

API DLL	Int WINAPI IBSU_WSQEncodeMem(const BYTE *image, const int width, const int height, cons tint pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText, BYTE **compressed Data, int *compressedLength)

- ***Description***

  WSQ compresses grayscale fingerprint image.

- ***Parameters***

Parameter	Description

*image	[in] Original image
width	[in] Width of original image (in pixels)
height	[in] Height of original image (in pixels)
pitch	[in] Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
bitsPerPixel	[in] Bits per pixel of original image
pixelPerInch	[in] Pixel per inch of orginal image

bitRate	[in] Determines the amount of lossy compression
	Suggested settings:
	bitRate = 2.25 yields around 5:1 compression
	bitRate = 0.75 yields around 15:1 compression

*commentText	[in] Comment to write compressed data
**compressedData	[out] Pointer of image which is compressed from orginal image by WSQ compression. This pointer is deallocated by IBSU_FreeMemory() after using it
*compressedLength	[out] Length of image which is compressed from original image by WSQ compression

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *

1.1.3.26) IBSU_WSQEncodeToFile

- ***Prototype***

API DLL	Int WINAPI IBSU_WSQEncodeToFile(LPCSTR filePath, const BYTE *image, const int width, const int height, cons tint pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText)

- ***Description***

Save WSQ compresses grayscale fingerprint image to specific file path.

- ***Parameters***

Parameter	Description

filePath	[in] File path to save image which is compressed from original image by WSQ compression
*image	[in] Original image
width	[in] Width of original image (in pixels)
height	[in] Height of original image (in pixels)
pitch	[in] Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
bitsPerPixel	[in] Bits per pixel of original image
pixelPerInch	[in] Pixel per inch of orginal image

bitRate	[in] Determines the amount of lossy compression
	Suggested settings:
	bitRate = 2.25 yields around 5:1 compression
	bitRate = 0.75 yields around 15:1 compression

*commentText	[in] Comment to write compressed data

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *

1.1.3.27) IBSU_WSQDecodeMem

- ***Prototype***

API DLL	Int WINAPI IBSU_WSQDecodeMem(const BYTE *compressedImage, const int compressedLength, BYTE **decompressedImage, int *outWidth, int *outHeight, int *outPitch, int *outBitsPerPixel, int *outPixelPerInch)

- ***Description***

Decompress a WSQ-encoded grayscale fingerprint image.

- ***Parameters***

Parameter	Description
*compressedImage	[in] WSQ-encoded image
compressedLength	[in] Length of WSQ-encoded image

**decompressedImag e	[out] Pointer of image which is decompressed from WSQ-encoded image. This pointer is deallocated by IBSU_FreeMemory() after using it
*outWidth	[out] Width of decompressed image (in pixels)
*outHeight	[out] Height of decompressed image (in pixels)
*outPitch	[out] Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
*outBitsPerPixel	[out] Bits per pixel of decompressed image
*outPixelPerInch	[out] Pixel per inch of decompressed image

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h *

1.1.3.28) IBSU_WSQDecodeFromFile

- ***Prototype***

API DLL	Int WINAPI IBSU_WSQDecodeFromFile(LPCSTR filePath, BYTE **decompressedImage, int *outWidth, int *outHeight, int outpitch, int *outBitsPerPixel, int *outPixelPerInch)

- ***Description***

Decompress a WSQ-encoded grayscale fingerprint image from specific file path.

- ***Parameters***

Parameter	Description
filePath	[in] File path of WSQ-encoded image
**decompressedImag e	[out] Pointer of image which is decompressed from WSQ-encoded image. This pointer is deallocated by IBSU_FreeMemory() after using it
*outWidth	[out] Width of decompressed image (in pixels)
*outHeight	[out] Height of decompressed image (in pixels)
*outPitch	[out] Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line

	order.
*outBitsPerPixel	[out] Bits per pixel of decompressed image
*outPixelPerInch	[out] Pixel per inch of decompressed image

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h [*]

1.1.3.29) IBSU_FreeMemory

- ***Prototype***

API DLL	Int WINAPI IBSU_FreeMemory(void *memblock)

- ***Description***

  Release the allocated memory block on the internal heap of library. This is obtained by IBSU_WSQEncodeMem(), IBSU_WSQDecodeMem(), IBSU_WSQDecodeFromFile() and other API functions

- ***Parameters***

Parameter	Description
memblock	[in] Previously allocated memory block to be freed

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h [†]

1.1.3.30) IBSU_SavePngImage

- ***Prototype***

API DLL	Int WINAPI IBSU_SavePngImage (LPCSTR filepath, const BYTE *image, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)

- ***Description***

Save fingerprint image in png format.

- ***Parameters***

Parameter	Description
filePath	[in] File path to save png
*image	[in] Point to raw image data (background color is black)
Width	[in] Image width
Height	[in] Image height
Pitch	[in] Image line pitch (Positive value indicate top down line order, Negative value mean bottom up line order)
resX	[in] Image horizontal resolution (in PPI)
resY	[in]Image vertical resolution (in PPI)

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.31) IBSU_SaveJP2Image

- ***Prototype***

API DLL	Int WINAPI IBSU_SaveJP2Image (LPCSTR filepath, const BYTE *image, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY, const int fQuality)

- ***Description***

Save fingerprint image in JPEG-2000 format.

- ***Parameters***

Parameter	Description
filePath	[in] File path to save jp2
*image	[in] Point to raw image data (background color is black)
Width	[in] Image width
Height	[in] Image height
Pitch	[in] Image line pitch (Positive value indicate top down line order, Negative value mean bottom up line order)

resX	[in] Image horizontal resolution (in PPI)
resY	[in]Image vertical resolution (in PPI)
fQuality	[in] Quality level for JPEG2000, he valid range is between 0 and 100

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.32) IBSU_CombineImage

- ***Prototype***

API DLL	Int WINAPI IBSU_CombineImage (const IBSU_ImageData inImage1, const IBSU_ImageData inImage2 ,IBSU_CombineImageWhichHand whichHand , IBSU_ImageData *ouImage)

- ***Description***

Combine two images (2 flat fingers) into a single image (left/right hands).

- ***Parameters***

Parameter	Description
inImage1	[in] Pointer to IBSU_ImageData (index and middle finger)
inImage2	[in] Pointer to IBSU_ImageData (ring and little finger)
whichHand	[in] Information of left or right hand
*ouImage	[out] Pointer to IBSU_ImageData (1600 x 1500 fixed size image)

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.33) IBSU_GetOperableBeeper

- ***Prototype***

API DLL	Int WINAPI IBSU_GetOperableBeeper (const int handle, IBSU_BeeperType *pBeeperType)

- ***Description***

Get characteristics of operable Beeper on a device.

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*pBeeperType	[out] Pointer to variable that will receive type of Beeper.

- ***IBSU_ BeeperType Enumerations***

```
/* No Beeper field. */
ENUM_IBSU_BEEPER_TYPE_NONE,
/* Monotone type. */
ENUM_IBSU_BEEPER_TYPE_MONOTONE
```

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.34) IBSU_SetBeeper

- ***Prototype***

API DLL	Int WINAPI IBSU_SetBeeper (const int handle, const IBSU_BeepPattern beepPattern, const DWORD soundTone, const DWORD duration, const DWORD reserved_1, const DWORD reserved_2)

- ***Description***

  Set the value of Beeper on a device.

- ***Parameters***

Parameter	Description
handle	[in] Device handle
beepPattern	[in] Pattern of beep
soundTone	[in] The frequency of the sound, in specific value. The parameter must be in the range 0 through 2
duration	[in] The duration of the sound, in 25 miliseconds. The parameter must be in the range 1 through 200 at ENUM_IBSU_BEEP_PATTERN_GENERIC, in the range 1 through 7 at ENUM_IBSU_BEEP_PATTERN_REPEAT.
reserved_1	[in] Reserved, If you set beepPattern to ENUM_IBSU_BEEP_PATTERN_REPEAT reserved_1 can use the sleep time after duration of the sound, in 25 miliseconds.
reserved_2	[in] Reserved, If you set beepPattern to ENUM_IBSU_BEEP_PATTERN_REPEAT reserved_2 can use the operation (start/stop of pattern repeat), 1 to start; 0 to stop.

- ***IBSU_ BeepPattern Enumerations***

```
ENUM_IBSU_BEEP_PATTERN_GENERIC,
```

ENUM_IBSU_BEEP_PATTERN_REPEAT

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.35) IBSU_CombineImageEx

- ***Prototype***

API DLL	Int WINAPI IBSU_CombineImageEx (const IBSU_ImageData inImage1, const IBSU_ImageData inImage2 ,IBSU_CombineImageWhichHand whichHand , IBSU_ImageData *ouImage, IBSU_ImageData *pSegmentImageArray, IBSU_SegmentPosition *pSegmentPositionArray, int *pSegmentImageArrayCount)

- ***Description***

  Combine two images (2 flat fingers) into a single image (left/right hands) and return segment information as well.

- ***Parameters***

Parameter	Description
inImage1	[in] Pointer to IBSU_ImageData (index and middle finger)
inImage2	[in] Pointer to IBSU_ImageData (ring and little finger)
whichHand	[in] Information of left or right hand
*ouImage	[out] Pointer to IBSU_ImageData (1600 x 1500 fixed size image)
pSegmentImageArray	Pointer to array of four structures that will receive individual finger image segments from output image. The buffers in these structures point to internal image buffers; the data should be copied to application buffers if desired for future processing.
pSegmentPositionArray	Pointer to array of four structures that will receive position data for individual fingers split from output image
pSegmentImageArrayCount	Pointer to variable that will receive number of finger images split from output image

- ***Returns***

Return Value	Description

0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.36) IBSU_CheckWetFinger

- ***Prototype***

API DLL	Int WINAPI IBSU_CheckWetFinger (const int handle, const IBSU_ImageData inImage)

- ***Description***

  Check if the image is wet or not.

- ***Parameters***

Parameter	Description
handle	[in] Device handle
inImage	[in] Pointer to IBSU_ImageData

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.37) IBSU_GetImageWidth

- ***Prototype***

API DLL	Int WINAPI IBSU_GetImageWidth (const int handle, const IBSU_ImageData inImage, int *Width_MM)

- ***Description***

  Get the width of input image by milli-meter(mm).

- ***Parameters***

Parameter	Description
handle	[in] Device handle
inImage	[in] Pointer to IBSU_ImageData
Width_MM	[out] width of inImage by milli-meter(mm)

- ***Returns***

Return Value	Description

0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.38) IBSU_IsWritableDirectory

- ***Prototype***

API DLL	Int WINAPI IBSU_IsWritableDirectory (LPCSTR dirpath, BOOL needCreateSubFolder)

- ***Description***

  Check whether a directory is writable

- ***Parameters***

Parameter	Description
dirpath	[in] Directory path
needCreateSubFolder	[in] Check whether need to create subfolder into the directory path

- ***Returns***

Return Value	Description
0	A directory is writable.
< 0	The error code as defined in IBScanUltimateApi_err.h IBSU_ERR_CHANNEL_IO_WRITE_FAILED: Directory does not writable.

1.1.3.39) IBSU_ GenerateDisplayImage

- ***Prototype***

API DLL	int WINAPI IBSU_GenerateDisplayImage(const BYTE *pInImage, const int inWidth, const int inHeight, BYTE *outImage, const int outWidth, const int outHeight, const BYTE outBkColor, const IBSU_ImageFormat outFormat, const int outQualityLevel, const BOOL outVerticalFlip)

- ***Description***

Generate scaled image in various formats for fast image display on canvas. You can use instead of IBSU_GenerateZoomOutImageEx()

- ***Parameters***

Paramet er	Description
*pInImage	[out] Original grayscale image data.
inWidth	[in] Width of input image.
inHeight	[in] Height of input image.
*outImage	[out] Pointer to buffer that will receive output image. This buffer must hold at least 'outWidth' x 'outHeight' x 'bitsPerPixel' bytes.
outWidth	[in] Width of output image.
outHeight	[in] Height of output image.
outBkColor	[in] Background color of output image.
outFormat	[in] IBSU_ImageFormat of output image.
outQualityLevel	[in] Image quality of output image. The parameter must be in the range 0 through 2
outVerticalFlip	[in] Enable/disable vertical flip of output image.

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.3.40) IBSU_ AddFingerImage

- ***Prototype***

API DLL	int WINAPI IBSU_AddFingerImage(const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const BOOL flagForce);

- ***Description***
This function adds a finger image to the buffer of IBScanUltimate for the fingerprint duplicate. The position of buffer should be designated by “fIndex” argument with bit-patterned values defined in “IBScanUltimateApi_Def.h”.
Currently, single finger image types(“ENUM_IBSU_ROLL_SINGLE_FINGER”, “ENUM_FLAT_SINGLE_FINGER”) are used only, other types are not supported.
In case user wants to update a buffer of the position but it is already used once, “flagForce=true” can be used to update the buffer. If “flagForce=false”, the buffer is not updated.
Or it is able to update after initializing the buffer with IBSU_RemoveFingerImage.

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
image	[in] Result image

fIndex	[in] Finger Index defiend in IBScanUltimateApi_Def.h.
	IBSU_FINGER_LEFT_LITTLE = 0x00000001
	IBSU_FINGER_LEFT_RING = 0x00000002
	IBSU_FINGER_LEFT_MIDDLE = 0x00000004
	IBSU_FINGER_LEFT_INDEX = 0x00000008
	IBSU_FINGER_LEFT_THUMB = 0x00000010
	IBSU_FINGER_RIGHT_THUMB = 0x00000020
	IBSU_FINGER_RIGHT_INDEX = 0x00000040
	IBSU_FINGER_RIGHT_MIDDLE = 0x00000080
	IBSU_FINGER_RIGHT_RING = 0x00000100
	IBSU_FINGER_RIGHT_LITTLE = 0x00000200

imageType	[in] Type of finger, roll or flat. IBSU_ImageType enumeration defined in

	IBScanUltimateApi.h /* Unspecified type. */ ENUM_IBSU_TYPE_NONE, /* One-finger rolled fingerprint. */ ENUM_IBSU_ROLL_SINGLE_FINGER, /* One-finger flat fingerprint. */ ENUM_IBSU_FLAT_SINGLE_FINGER, /* Two-finger flat fingerprint. */ ENUM_IBSU_FLAT_TWO_FINGERS, /* Four-finger flat fingerprint. */ ENUM_IBSU_FLAT_FOUR_FINGERS, /* Three-finger flat fingerprint. */ ENUM_IBSU_FLAT_THREE_FINGERS
flagForce	It decides to force writing the fingerimage. If this is is true, it overwrites with the fingerimage. If this is TRUE, the designated buffer can be overwritten.

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

.

1.1.3.41) IBSU_ RemoveFingerImage

- ***Prototype***

API DLL	int WINAPI IBSU_RemoveFingerImage(const int handle, const DWORD fIndex)

- ***Description***
  This function removes finger images selected by the “fIndex” argument.
  One and more finger indexes can be designated.

- ***Parameter***

Parameter	Description

handle	[in] Device handle obtained by IBSU_OpenDevice().
fIndex	[in] Finger Index defiend in IBScanUltimateApi_Def.h. IBSU_FINGER_LEFT_LITTLE = 0x00000001 IBSU_FINGER_LEFT_RING = 0x00000002 IBSU_FINGER_LEFT_MIDDLE = 0x00000004 IBSU_FINGER_LEFT_INDEX = 0x00000008 IBSU_FINGER_LEFT_THUMB = 0x00000010 IBSU_FINGER_RIGHT_THUMB = 0x00000020 IBSU_FINGER_RIGHT_INDEX = 0x00000040 IBSU_FINGER_RIGHT_MIDDLE = 0x00000080 IBSU_FINGER_RIGHT_RING = 0x00000100 IBSU_FINGER_RIGHT_LITTLE = 0x00000200 and mix combinations : IBSU_FINGER_LEFT_HAND = (IBSU_FINGER_LEFT_INDEX | IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_RING | IBSU_FINGER_LEFT_LITTLE) IBSU_FINGER_RIGHT_HAND = (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE | IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE) IBSU_FINGER_BOTH_THUMBS = (IBSU_FINGER_RIGHT_THUMB | IBSU_FINGER_LEFT_THUMB) IBSU_FINGER_ALL = (IBSU_FINGER_LEFT_HAND | IBSU_FINGER_RIGHT_HAND | IBSU_FINGER_BOTH_THUMBS) IBSU_FINGER_LEFT_LITTLE_RING = (IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING) IBSU_FINGER_LEFT_MIDDLE_INDEX = (IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_INDEX) IBSU_FINGER_RIGHT_INDEX_MIDDLE = (IBSU_FINGER_RIGHT_INDEX |

	IBSU_FINGER_RIGHT_MIDDLE) IBSU_FINGER_RIGHT_RING_LITTLE = (IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

.

1.1.3.42) IBSU_ IsFingerDuplicated

- ***Prototype***

API DLL	int WINAPI IBSU_IsFingerDuplicated(const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const int securityLevel, DWORD *pMatchedPosition)

- ***Description***

This function compares a finger image with the registered images designated by "fIndex" argument.
Matching threshold is set with "SecurityLevel", it can be set from 1 to 7.
7 is the highest threshold.
The matching result is returned with "pMatchedPosition" in bit-pattern. If it matches, bit-pattern of FingerIndex values is returned. The value indicates the finger is matched to the positions of buffer of IBScanUltimate. If there is nothing matched, '0' is returned.

One and more finger positions can be designated. In case multiple positions are designated, and one of the matches is TRUE, it returns TRUE.

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
image	[in] Result image
fIndex	[in] Finger Index defiend in IBScanUltimateApi_Def.h.

	IBSU_FINGER_LEFT_LITTLE = 0x00000001 IBSU_FINGER_LEFT_RING = 0x00000002 IBSU_FINGER_LEFT_MIDDLE = 0x00000004 IBSU_FINGER_LEFT_INDEX = 0x00000008 IBSU_FINGER_LEFT_THUMB = 0x00000010 IBSU_FINGER_RIGHT_THUMB = 0x00000020 IBSU_FINGER_RIGHT_INDEX = 0x00000040 IBSU_FINGER_RIGHT_MIDDLE = 0x00000080 IBSU_FINGER_RIGHT_RING = 0x00000100 IBSU_FINGER_RIGHT_LITTLE = 0x00000200 and mix combinations : IBSU_FINGER_LEFT_HAND = (IBSU_FINGER_LEFT_INDEX | IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_RING | IBSU_FINGER_LEFT_LITTLE) IBSU_FINGER_RIGHT_HAND = (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE | IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE) IBSU_FINGER_BOTH_THUMBS = (IBSU_FINGER_RIGHT_THUMB | IBSU_FINGER_LEFT_THUMB) IBSU_FINGER_ALL = (IBSU_FINGER_LEFT_HAND | IBSU_FINGER_RIGHT_HAND | IBSU_FINGER_BOTH_THUMBS) IBSU_FINGER_LEFT_LITTLE_RING = (IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING) IBSU_FINGER_LEFT_MIDDLE_INDEX = (IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_INDEX) IBSU_FINGER_RIGHT_INDEX_MIDDLE = (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE) IBSU_FINGER_RIGHT_RING_LITTLE = (IBSU_FINGER_RIGHT_RING |

	IBSU_FINGER_RIGHT_LITTLE)
imageType	[in] Type of finger, roll or flat. IBSU_ImageType enumeration defined in IBScanUltimateApi.h /* Unspecified type. */ ENUM_IBSU_TYPE_NONE, /* One-finger rolled fingerprint. */ ENUM_IBSU_ROLL_SINGLE_FINGER, /* One-finger flat fingerprint. */ ENUM_IBSU_FLAT_SINGLE_FINGER, /* Two-finger flat fingerprint. */ ENUM_IBSU_FLAT_TWO_FINGERS, /* Four-finger flat fingerprint. */ ENUM_IBSU_FLAT_FOUR_FINGERS, /* Three-finger flat fingerprint. */ ENUM_IBSU_FLAT_THREE_FINGERS
securityLevel	[in] Threshold for match, from 1 to 7. Higher value the more extractions of finger are required
*pMatchedPosition	[out] Result of match. If it matches, bit-pattern of FingerIndex value is returned The value indicates the finger is matched to the positions of the buffer of IBScanUltimate. If there is nothing matched, ‘0’ is returned.

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

.

1.1.3.43) IBSU_ IsValidFingerGeometry

- ***Prototype***

API DLL	int WINAPI IBSU_IsValidFingerGeometry(const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType,

	BOOL *pValid)

- ***Description***
  This function checks the shape of fingers in the image by the "fIndex" argument, and returns the result of match in Boolean type.
  In case of 4-finger it can identify left or right hand, and in case of 2-finger it can identify "little-ring" or "index-middle".

- ***Parameter***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice().
image	[in] Result image of 4-fingers or 2-fingers

fIndex	[in] Finger Index defiend in IBScanUltimateApi_Def.h.
	
	Designation of finger position.
	/* for 4-finger */
	IBSU_FINGER_LEFT_HAND = (IBSU_FINGER_LEFT_INDEX | IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_RING | IBSU_FINGER_LEFT_LITTLE)
	
	IBSU_FINGER_RIGHT_HAND = (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE | IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)
	
	/* for 2-finger */
	IBSU_FINGER_LEFT_LITTLE_RING = (IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING)
	
	IBSU_FINGER_LEFT_MIDDLE_INDEX = (IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_INDEX)
	
	IBSU_FINGER_RIGHT_INDEX_MIDDLE = (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE)
	
	IBSU_FINGER_RIGHT_RING_LITTLE = (IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)

imageType	[in] Type of finger, roll or flat. IBSU_ImageType enumeration defined in IBScanUltimateApi.h

	/* Unspecified type. */ ENUM_IBSU_TYPE_NONE, /* One-finger rolled fingerprint. */ ENUM_IBSU_ROLL_SINGLE_FINGER, /* One-finger flat fingerprint. */ ENUM_IBSU_FLAT_SINGLE_FINGER, /* Two-finger flat fingerprint. */ ENUM_IBSU_FLAT_TWO_FINGERS, /* Four-finger flat fingerprint. */ ENUM_IBSU_FLAT_FOUR_FINGERS, /* Three-finger flat fingerprint. */ ENUM_IBSU_FLAT_THREE_FINGERS
pValid	[out] “TRUE” if match is succussful, or “FALSE” is returned.

- ***Return***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

2.1.4.Client Window Interface Functions

1.1.4.1) IBSU_CreateClientWindow (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_CreateClientWindow (const int handle, const IBSU_HWD hWindow, const DWORD left, const DWORD top, const DWORD right, const DWORD bottom)

- ***Description***

Make a user-defined fingerprint window (window size and location).

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
hWindow	[in] Windows handle to draw
left	[in] Rectangle coordinates to draw (top, bottom, left, right)
top	[in] Rectangle coordinates to draw (top, bottom, left, right)
right	[in] Rectangle coordinates to draw (top, bottom, left, right)
bottom	[in] Rectangle coordinates to draw (top, bottom, left, right)

- ***IBSU_ HWD Definions***

```
#ifdef _WINDOWS
#define IBSU_HWND     HWND
#define IBSU_RECT     RECT
#else
#define IBSU_HWND     void *
#define IBSU_RECT     void *
#endif
```

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.2) IBSU_DestroyClientWindow (Windows only)

- ***Prototype***

API DLL	int IBSU_DestroyClientWindow (const int handle, const BOOL clearExistingInfo)

- ***Description***

Release a user-defined window.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
clearExistingInfo	[in] Clear the existing display property and overlay test information.

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.3) IBSU_GetClientWindowProperty (Windows only)

- ***Prototype***

API DLL	int IBSU_GetClientWindowProperty (const int handle, const IBSU_ClientWindowPropertyId propertyId, LPSTR propertyValue)

- ***Description***

Get user-defined window properties.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
propertyId	[in] Property identifier to set value
propertyValue	[out] String returning the property's value. (Memory must be provided by caller)

- ***IBSU_ClientWindowPropertyId Enumerations (Settable)***

```
/* Background color of display window.   The valid range is between 0x00000000 and
0xFFFFFFFF, inclusive, with the default of 0x00D8E9EC (the button face color on
Windows). [Get and set.] */
ENUM_IBSU_WINDOW_PROPERTY_BK_COLOR,
/* Indicates whether guide line should be drawn for rolling print capture (TRUE or FALSE).
The default is TRUE.   [Get and set.] */
ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE,
/* Draw arrow to display invalid area (TRUE or FALSE).   The default is FALSE.   [Get and
set.] */
ENUM_IBSU_WINDOW_PROPERTY_DISP_INVALID_AREA,
/* Thickness of ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE The valid range
is between 1 and 6 pixels, inclusive, with the default of 2 pixels.   [Get and set.] */
ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_WIDTH,
```

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.4) IBSU_SetClientDisplayProperty (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_SetClientDisplayProperty(const int handle, const IBSU_ClientWindowPropertyId propertyId, LPCSTR propertyValue)

- ***Description***

Set a user-define window property.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
propertyId	[in] Property identifier to set value
propertyValue	[out] String returning property value. (Memory must be provided by caller)

- ***IBSU_ClientWindowPropertyId Enumerations (Settable)***

  /* Background color of display window. The valid range is between 0x00000000 and 0xFFFFFFFF, inclusive, with the default of 0x00D8E9EC (the button face color on Windows). [Get and set.] */

  ENUM_IBSU_WINDOW_PROPERTY_BK_COLOR,

  /* Indicates whether guide line should be drawn for rolling print capture (TRUE or FALSE). The default is TRUE. [Get and set.] */

  ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE,

  /* Draw arrow to display invalid area (TRUE or FALSE). The default is FALSE. [Get and set.] */

  ENUM_IBSU_WINDOW_PROPERTY_DISP_INVALID_AREA,

  /* Get the scale of the display image on client window, as a floating point value. */

  ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR,

  /* Get the left margin of the displayed image in relation to the client window, as an integer. */

  ENUM_IBSU_WINDOW_PROPERTY_LEFT_MARGIN,

  /* Get the top margin of the displayed image in relation to the client window, as an integer. */

  ENUM_IBSU_WINDOW_PROPERTY_TOP_MARGIN,

  /* Thickness of ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE The valid range is between 1 and 6 pixels, inclusive, with the default of 2 pixels. [Get and set.] */

  ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_WIDTH,

  /* Get the extended scale of the display image on client window, as a integer value. */

  ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR_EX,

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.5) IBSU_SetClientWindowOverlayText (Windows only) (Deprecated)

- ***Prototype***

API DLL	Int WINAPI IBSU_SetClientWindowOverlayText (const int handle, const char *fontName, const int fontSize, const BOOL fontBold, const char *text, const int posX, cons tint posY, const

	DWRD textColor)

- ***Description***

  Set the text property on a user-defined window.

- ***Parameters***

Parameter	Description
handle	[in] Device handle obtained by IBSU_OpenDevice()
*fontName	[in] font name for display
fontsize	[in] font size for display
fontBold	[in] font bold for display
*text	[in] string for display
posX	[in] X coordinate of text for display
posY	[in] Y coordinate of text for display
textColor	[in] string color for display

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.6) IBSU_ShowOverlayObject (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_ShowOverlayObject (const int handle, const int overlayHandle, const BOOL show);

- ***Description***

  Show or hide an overlay object

- ***Parameters***

Parameter	Description
handle	[in] Device handle
overlayHandle	[in] Overlay handle obtained by overlay functions
show	[in] Overlay will be shown/hidden on client window

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.7) IBSU_ShowAllOverlayObject (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_ShowAllOverlayObject (const int handle, const BOOL show);

- ***Description***

Show all overlay objects

- ***Parameters***

Parameter	Description
handle	[in] Device handle
show	[in] Overlay will be shown/hidden on client window

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.8) IBSU_RemoveOverlayObject (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_RemoveOverlayObject (const int handle, const int overlayHandle);

- ***Description***

Remove an overlay object.

- ***Parameters***

Parameter	Description
handle	[in] Device handle

overlayHandle	[in] Overlay handle obtained by overlay functions

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.9) IBSU_RemoveAllOverlayObject (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_RemoveAllOverlayObject (const int handle);

- ***Description***

Remove all overlay objects.

- ***Parameters***

Parameter	Description
handle	[in] Device handle

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.10) IBSU_AddOverlayText (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_AddOverlayText(const int handle, int *pOverlayHandle, const char *fontName, const int fontSize, const BOOL fontBold, const char *text, const int posX, const int posY, const DWORD textColor

	);

- ***Description***

  Add an overlay text for display on window.

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*pOverlayHandle	[out] Function returns overlay handle to be used for client windows functions call
*fontName	[in] Name of font.
fontSize	[in] Font size.
fontBold	[in] Indicates whether font is bold.
*text	[in] Text for display on window
posX	[in] X coordinate of text for display on window
posY	[in] Y coordinate or test for display on window
textColor	[in] Text color

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.11) IBSU_ModifyOverlayText (Windows only)

- ***Prototype***

<table><tr><td>API DLL</td><td><pre>int WINAPI IBSU_ModifyOverlayText(
        const int   handle,
        int   OverlayHandle,
        const char   *fontName,
        const int   fontSize,
        const BOOL   fontBold,
        const char   *text,
        const int   posX,
        const int   posY,
        const DWORD   textColor
    );</pre></td></tr></table>

- ***Description***

  Modify an existing overlay text for display on window

- ***Parameters***

Parameter	Description
handle	[in] Device handle
OverlayHandle	[in] Handle of overlay to modify.
*fontName	[in] Name of font.
fontSize	[in] Font size.
fontBold	[in] Indicates whether font is bold.
*text	[in] Text for display on window
posX	[in] X coordinate of text for display on window
posY	[in] Y coordinate or test for display on window
textColor	[in] Text color

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.12) IBSU_AddOverlayLine (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_AddOverlayLine(const int handle, int *pOverlayHandle, const int x1, const int y1, const int x2, const int y2, const int lineWidth const DWORD lineColor);

- ***Description***

  Add an overlay line for display on window

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*pOverlayHandle	[out] Function returns overlay handle to be used for client windows functions calls
x1	[in] X coordinate of start point of line
y1	[in] Y coordinate of start point of line
x2	[in] X coordinate of end point of line
y2	[in] Y coordinate of end point of line
lineWidth	[in] line width
lineColor	[in] line color

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.13) IBSU_ModifyOverlayLine (Windows only)

- ***Prototype***

API DLL	Prototype
API DLL	int WINAPI IBSU_ModifyOverlayLine(const int handle, const int OverlayHandle, const int x1, const int y1, const int x2, const int y2, const int lineWidth const DWORD lineColor);

- ***Description***

Modify an existing line for display on window

- ***Parameters***

Parameter	Description
handle	[in] Device handle

OverlayHandle	[in] Handle of overlay to modify
x1	[in] X coordinate of start point of line.
y1	[in] Y coordinate of start point of line.
x2	[in] X coordinate of end point of line.
y2	[in] Y coordinate of end point of line.
lineWidth	[in] line width
lineColor	[in] line color

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.14) IBSU_AddOverlayQuadrangle (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_AddOverlayQuadrangle(const int handle, int *pOverlayHandle, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3, const int x4, const int y4, const int lineWidth const DWORD lineColor);

- ***Description***

Add an overlay quadrangle for display on window

- ***Parameters***

Parameter	Description
handle	[in] Device handle

*pOverlayHandle	[out] Function returns overlay handle to be used for client windows functions calls
x1	[in] X coordinate of 1st vertex of quadrangle
y1	[in] Y coordinate of 1st vertex of quadrangle
x2	[in] X coordinate of 2nd vertex of quadrangle
y2	[in] Y coordinate of 2nd vertex of quadrangle
x3	[in] X coordinate of 3rd vertex of quadrangle
y3	[in] Y coordinate of 3rd vertex of quadrangle
x4	[in] X coordinate of 4th vertex of quadrangle
y4	[in] Y coordinate of 4th vertex of quadrangle
lineWidth	[in] line width
lineColor	[in] line color

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.15) IBSU_ModifyOverlayQuadrangle (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_ModifyOverlayQuadrangle(const int handle, const int OverlayHandle, const int x1, const int y1, const int x2, const int y2, const int x3, const int y3, const int x4, const int y4, const int lineWidth const DWORD lineColor);

- ***Description***

Modify an existing quadrangle for display on window

- ***Parameters***

Parameter	Description
handle	[in] Device handle
overlayHandle	[in] Handle of overlay to modify.
x1	[in] X coordinate of 1st vertex of quadrangle
y1	[in] Y coordinate of 1st vertex of quadrangle
x2	[in] X coordinate of 2nd vertex of quadrangle
y2	[in] Y coordinate of 2nd vertex of quadrangle
x3	[in] X coordinate of 3rd vertex of quadrangle
y3	[in] Y coordinate of 3rd vertex of quadrangle
x4	[in] X coordinate of 4th vertex of quadrangle
y4	[in] Y coordinate of 4th vertex of quadrangle
lineWidth	[in] line width
lineColor	[in] line color

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.16) IBSU_AddOverlayShape (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_AddOverlayShape(const int handle, int *pOverlayHandle, const IBSU_OverlayShapePattern shapePattern, const int x1, const int y1, const int x2, const int y2, const int lineWidth, const DWORD lineColor, const int reserved_1,

	const int reserved_2);

- ***Description***

  Add an overlay shape for display on window

- ***Parameters***

Parameter	Description
handle	[in] Device handle
*pOverlayHandle	[out] Function returns overlay handle to be used for client windows functions calls
shapePattern	[in] Pattern of shape
x1	[in] X coordinate of start point of overlay shape
y1	[in] Y coordinate of start point of overlay shape
x2	[in] X coordinate of end point of overlay shape
y2	[in] Y coordinate of end point of overlay shape
lineWidth	[in] line width
lineColor	[in] line color
reserved_1	[in] Reserved
reserved_2	[in] Reserved

- ***IBSU_OverlayShapePattern Enumerations***

  ENUM_IBSU_OVERLAY_SHAPE_RECTANGLE,

  ENUM_IBSU_OVERLAY_SHAPE_ELLIPSE,

  ENUM_IBSU_OVERLAY_SHAPE_CROSS,

  ENUM_IBSU_OVERLAY_SHAPE_ARROW

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.17) IBSU_ModifyOverlayShape (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_ModifyOverlayShape(const int handle, int OverlayHandle,

	const IBSU_OverlayShapePattern shapePattern, const int x1, const int y1, const int x2, const int y2, const int lineWidth, const DWORD lineColor, const int reserved_1, const int reserved_2);

- ***Description***

  Modify an overlay shape for display on window

- ***Parameters***

Parameter	Description
handle	[in] Device handle
OverlayHandle	[in] Overlay handle to modify
shapePattern	[in] Pattern of shape
x1	[in] X coordinate of start point of overlay shape
y1	[in] Y coordinate of start point of overlay shape
x2	[in] X coordinate of end point of overlay shape
y2	[in] Y coordinate of end point of overlay shape
lineWidth	[in] line width
lineColor	[in] line color
reserved_1	[in] Reserved
reserved_2	[in] Reserved

- ***IBSU_OverlayShapePattern Enumerations***

ENUM_IBSU_OVERLAY_SHAPE_RECTANGLE,

ENUM_IBSU_OVERLAY_SHAPE_ELLIPSE,

ENUM_IBSU_OVERLAY_SHAPE_CROSS,

ENUM_IBSU_OVERLAY_SHAPE_ARROW

- ***Returns***

Return Value	Description
0	Function completed successfully.

< 0	The error code as defined in IBScanUltimateApi_err.h

1.1.4.18) IBSU_RedrawClientWindow (Windows only)

- ***Prototype***

API DLL	int WINAPI IBSU_RedrawClientWindow(const int handle);

- ***Description***

Update the specified client window which is defined by IBSU_CreateClientWindow().

- ***Parameters***

Parameter	Description
handle	[in] Device handle

- ***Returns***

Return Value	Description
0	Function completed successfully.
< 0	The error code as defined in IBScanUltimateApi_err.h

2.1.5.Callback Interface Functions

1.1.5.1) IBSU_Callback()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_Callback) (const int deviceHandle, void *pContext)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK, called when communication with a device is interrupted.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.

1.1.5.2) IBSU_CallbackPreviewImage()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackPreviewImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE, called when a preview image is available.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
image	[out] Preview image data. This structure, including the buffer, is valid only within the function context. If required for later use, any data must be copied to another structure.

- ***IBSU_ ImageData Structure Definition***

```
typedef struct tagIBSU_ImageData
{
    /* Pointer to image buffer.   If this structure is supplied by a callback function, this pointer
    must not be retained; the data should be copied to an application buffer for any
    processing after the callback returns. */
    void                *Buffer;
    /* Image horizontal size (in pixels). */
    DWORD               Width;
    /* Image vertical size (in pixels). */
    DWORD               Height;
    /* Horizontal image resolution (in pixels/inch). */
    double              ResolutionX;
    /* Vertical image resolution (in pixels/inch). */
    double              ResolutionY;
    /* Image acquisition time, excluding processing time (in seconds). */
    double              FrameTime;
    /* Image line pitch (in bytes).   A positive value indicates top-down line order; a negative
    value indicates bottom-up line order. */
    int                 Pitch;
    /* Number of bits per pixel. */
    BYTE                BitsPerPixel;
    /* Image color format. */
    IBSU_ImageFormat    Format;
    /* Marks image as the final processed result from the capture.   If this is FALSE, the
    image is a preview image or a preliminary result. */
    BOOL                IsFinal;
    /* Threshold of image processing. */
    DWORD               ProcessThres;
}
```

1.1.5.3) IBSU_CallbackFingerCount()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackFingerCount) (const int deviceHandle, void *pContext, const IBSU_FingerCountState fingerCountState)

- ***Description***

Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT, called when the finger count changes.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
fingerCountState	[out] Finger count state

- ***IBSU_ FingerCountState Enumerations***

ENUM_IBSU_FINGER_COUNT_OK,

ENUM_IBSU_TOO_MANY_FINGERS,

ENUM_IBSU_TOO_FEW_FINGERS,

ENUM_IBSU_NON_FINGER

1.1.5.4) IBSU_CallbackFingerQuality()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackFingerQuality) (const int deviceHandle, void *pContext, const IBSU_FingerQualityState *pQualityArray, const int qualityArrayCount)

- ***Description***

Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY, called when a finger quality changes.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
*pQualityArray	[out] Array of finger qualities
qualityArrayCount	[out] Number of qualities in array

- ***IBSU_ FingerQualityState Enumerations***

ENUM_IBSU_FINGER_NOT_PRESENT,

ENUM_IBSU_QUALITY_GOOD,

ENUM_IBSU_QUALITY_FAIR,

ENUM_IBSU_QUALITY_POOR,

/* Finger position is not valid on top side. */

```
ENUM_IBSU_QUALITY_INVALID_AREA_TOP,
/* Finger position is not valid on left side. */
ENUM_IBSU_QUALITY_INVALID_AREA_LEFT,
/* Finger position is not valid on right side. */
ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT,
/* Finger position is not valid on bottom side. */
ENUM_IBSU_QUALITY_INVALID_AREA_BOTTOM
```

1.1.5.5) IBSU_CallbackDeviceCount()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackDeviceCount) (const int detectedDevices, void *pContext)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT, called when the number of detected devices changes.

- ***Parameters***

Parameter	Description
detectedDevices	[out] Number of detected devices
*pContext	[out] User context.

1.1.5.6) IBSU_CallbackInitProgress()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackInitProgress) (const int deviceIndex, void *pContext, const int progressValue)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS, called when the initialization progress changes for a device.

- ***Parameters***

Parameter	Description
deviceIndex	[out] Zero-based index of device
*pContext	[out] User context.
progressValue	[out] Initialization progress, as a percent, between 0 and 100,

	inclusive.

1.1.5.7) IBSU_CallbackTakingAcquisition()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackTakingAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)

- ***Description***

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION, called for a rolled print acquisition when the rolling should begin.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
imageType	[out] Type of image being acquired.

- ***IBSU_ImageType Enumerations***

```
/* Unspecified type. */
ENUM_IBSU_TYPE_NONE,
/* One-finger rolled fingerprint. */
ENUM_IBSU_ROLL_SINGLE_FINGER,
/* One-finger flat fingerprint. */
ENUM_IBSU_FLAT_SINGLE_FINGER,
/* Two-finger flat fingerprint. */
ENUM_IBSU_FLAT_TWO_FINGERS,
/* Four-finger flat fingerprint. */
ENUM_IBSU_FLAT_FOUR_FINGERS
```

1.1.5.8) IBSU_CallbackCompleteAcquisition()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackCompleteAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION, called for a rolled print acquisition when the rolling capture has completed.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
imageType	[out] Type of image being acquired.

- ***IBSU_ImageType Enumerations***

```
/* Unspecified type. */
ENUM_IBSU_TYPE_NONE,
/* One-finger rolled fingerprint. */
ENUM_IBSU_ROLL_SINGLE_FINGER,
/* One-finger flat fingerprint. */
ENUM_IBSU_FLAT_SINGLE_FINGER,
/* Two-finger flat fingerprint. */
ENUM_IBSU_FLAT_TWO_FINGERS,
/* Four-finger flat fingerprint. */
ENUM_IBSU_FLAT_FOUR_FINGERS
```

1.1.5.9) IBSU_CallbackResultImage() (Deprecated)

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackResultImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image, const IBSU_ImageType imageType, const IBSU_ImageData *pSplitImageArray, const int splitImageArrayCount)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE, called when the result image is available.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.

image	[out] Data of preview or result image. The buffer in this structure points to an internal image buffer; the data should be copied to an application buffer if desired for future processing.
imageType	[out] Image type.
*pSplitImageArray	[out] Array of four structures with data of individual finger images split from result image. The buffers in these structures point to internal image buffers; the data should be copied to application buffers if desired for future processing.
splitImageArrayCount	[out] Number of finger images split from result images.

- ***IBSU_ ImageData Structure Definition***

```
typedef struct tagIBSU_ImageData
{
    /* Pointer to image buffer.  If this structure is supplied by a callback function, this pointer
    must not be retained; the data should be copied to an application buffer for any
    processing after the callback returns. */
    void                *Buffer;
    /* Image horizontal size (in pixels). */
    DWORD                   Width;
    /* Image vertical size (in pixels). */
    DWORD                   Height;
    /* Horizontal image resolution (in pixels/inch). */
    double              ResolutionX;
    /* Vertical image resolution (in pixels/inch). */
    double              ResolutionY;
    /* Image acquisition time, excluding processing time (in seconds). */
    double              FrameTime;
    /* Image line pitch (in bytes).  A positive value indicates top-down line order; a negative
    value indicates bottom-up line order. */
    int                 Pitch;
    /* Number of bits per pixel. */
    BYTE                    BitsPerPixel;
    /* Image color format. */
    IBSU_ImageFormat  Format;
    /* Marks image as the final processed result from the capture.  If this is FALSE, the
    image is a preview image or a preliminary result. */
    BOOL                    IsFinal;
    /* Threshold of image processing. */
    DWORD                    ProcessThres;
  }
```

- ***IBSU_ImageType Enumerations***

```
/* Unspecified type. */
ENUM_IBSU_TYPE_NONE,
/* One-finger rolled fingerprint. */
ENUM_IBSU_ROLL_SINGLE_FINGER,
/* One-finger flat fingerprint. */
ENUM_IBSU_FLAT_SINGLE_FINGER,
/* Two-finger flat fingerprint. */
ENUM_IBSU_FLAT_TWO_FINGERS,
/* Four-finger flat fingerprint. */
ENUM_IBSU_FLAT_FOUR_FINGERS
```

1.1.5.10) IBSU_CallbackResultImageEx()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackResultImageEx) (const int deviceHandle, void *pContext, const int imageStatus, const IBSU_ImageData image, const IBSU_ImageType imageType, const int detectedFingerCount, const int segmentImageArrayCount, const IBSU_ImageData *pSegmentImageArray, const IBSU_SegmentPosition *pSegmentPositionArray)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE_EX, called when the result image is available, with extended information.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
imageStatus	[out] Status from result image acquisition.
image	[out] Data of preview or result image. The buffer in this structure points to an internal image buffer; the data should be copied to

	an application buffer if desired for future processing.
imageType	[out] Image type.
detectedFingerCount	[out] Number of detected fingers.
segmentImageArrayCount	[out] Number of finger images split from result images.
*pSegmentImageArray	[out] Array of structures with data of individual finger images split from result image. The buffers in these structures point to internal image buffers; the data should be copied to application buffers if desired for future processing.
*pSegmentPositionArray	[out] Array of structures with position data for individual fingers split from result image.

- ***IBSU_ ImageData Structure Definition***

```
typedef struct tagIBSU_ImageData
{
    /* Pointer to image buffer.  If this structure is supplied by a callback function, this pointer
    must not be retained; the data should be copied to an application buffer for any
    processing after the callback returns. */
    void                *Buffer;
    /* Image horizontal size (in pixels). */
    DWORD               Width;
    /* Image vertical size (in pixels). */
    DWORD               Height;
    /* Horizontal image resolution (in pixels/inch). */
    double              ResolutionX;
    /* Vertical image resolution (in pixels/inch). */
    double              ResolutionY;
    /* Image acquisition time, excluding processing time (in seconds). */
    double              FrameTime;
    /* Image line pitch (in bytes).  A positive value indicates top-down line order; a negative
    value indicates bottom-up line order. */
    int                 Pitch;
    /* Number of bits per pixel. */
    BYTE                BitsPerPixel;
    /* Image color format. */
    IBSU_ImageFormat    Format;
    /* Marks image as the final processed result from the capture.   If this is FALSE, the
    image is a preview image or a preliminary result. */
    BOOL                    IsFinal;
    /* Threshold of image processing. */
    DWORD                   ProcessThres;
  }
```

- ***IBSU_ImageType Enumerations***

```
/* Unspecified type. */
ENUM_IBSU_TYPE_NONE,
/* One-finger rolled fingerprint. */
ENUM_IBSU_ROLL_SINGLE_FINGER,
/* One-finger flat fingerprint. */
ENUM_IBSU_FLAT_SINGLE_FINGER,
/* Two-finger flat fingerprint. */
ENUM_IBSU_FLAT_TWO_FINGERS,
/* Four-finger flat fingerprint. */
ENUM_IBSU_FLAT_FOUR_FINGERS
```

- ***IBSU_ SegmentPosition Structure Definition***

```
typedef struct tagIBSU_ImageData
{
    /* X coordinate of starting point of the finger segment. */
    short x1;
    /* Y coordinate of starting point of the finger segment. */
    short y1;
    /* X coordinate of 1st corner of the finger segment. */
    short x2;
    /* Y coordinate of 1st corner of the finger segment. */
    short y2;
    /* X coordinate of 2nd corner of the finger segment. */
    short x3;
    /* Y coordinate of 2nd corner of the finger segment. */
    short y3;
    /* X coordinate of 3rd corner of the finger segment. */
    short x4;
    /* Y coordinate of 3rd corner of the finger segment. */
    short y4;
}
```

1.1.5.11) IBSU_CallbackClearPlatenAtCapture()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackClearPlatenAtCapture) (const int deviceHandle, void *pContext, const IBSU_PlatenState platenState)

- ***Description***

Callback for ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE, called when the platen was not clear when capture started or has since become clear.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
platenState	[out] Platen state.

- ***IBSU_PlatenState Enumerations***

ENUM_IBSU_PLATEN_CLEARD,

ENUM_IBSU_PLATEN_HAS_FINGERS

1.1.5.12) IBSU_CallbackAsyncOpenDevice()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackAsyncOpenDevice) (const int deviceIndex, void *pContext, const int deviceHandle, const int errorCode)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE, called asynchronous device initialization completes

- ***Parameters***

Parameter	Description
deviceIndex	[out] Zero-based index of device.
*pContext	[out] User context.

deviceHandle	[out] Handle for subsequent function calls.
errorCode	[out] Error that prevented initialization from completing.

1.1.5.13) IBSU_CallbackNotifyMessage()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackNotifyMessage) (const int deviceHandle, void *pContext, const int notifyMessage)

- ***Description***

Callback for ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE, called when a warning message is generated.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
notifyMessage	[out] Handle for subsequent function calls.
errorCode	[out] Status code as defined in IBScanUltimateApi_err

1.1.5.14) IBSU_CallbackKeyButtons()

- ***Prototype***

API DLL	typede void (CALLBACK *IBSU_CallbackKeyButtons) (const int deviceHandle, void *pContext, const int pressedKeyButtons)

- ***Description***

Callback for ENUM_IBSU_ESSENTIAL_EVENT_KEYBUTTON, called when the key button of device was chicked.

- ***Parameters***

Parameter	Description
deviceHandle	[out] Device handle obtained by IBSU_OpenDevice()
*pContext	[out] User context.
pressedKeyButtons	[out] The key button index which is pressed.

Support Contact Information:

www.integratedbiometrics.com

Integrated Biometrics, LLC

North American Office

Physical Address for Package Delivery
121 Broadcast Drive
Spartanburg SC 29303

For Mailings & Correspondence
PO Box 170938
Spartanburg, SC 29301

US & Canada
(864) 990-3711
Toll-free (888) 840-8034
Extension 1 – Company Directory
Extension 2 – Technical Support
Extension 3 – Sales Support
Extension 4 – Marketing
Extension 5 – Accounting
Extension 0 – Main Line

Sales & Pricing Inquiries
sales@integratedbiometrics.com

Terms & Conditions of a Sale

Terms & Conditions for Supplier Purchases

Sales Administration
marci.bowers@integratedbiometrics.com

Technical Support
technical@integratedbiometrics.com

South Korean Office

Physical Address and Mailing Address
#910 Suntech-City1, 513-15
Sangdaewon 1-dong Jungwon-gu
Seongnam-si, Gyeonggi-do
Republic of Korea

Phone
+82-31-777-2207

Sales Administration
everun@ibkr.co.kr