IBScanUltimate API Manual for C/C++

4.2.1

Generated by Doxygen 1.9.6

Chapter 1

IBScanUltimate API Manual for C/C++

1.1 About API Manual

Version

4.2.1

Date

July.18 2025

1.2 Index

Chapter 2

API Function Lists

2.1 Device

- API - Device - Open/Close
- API - Device - Infomation
- API - Device - Property
- API - Device - Image Aquisition
- API - Device - General

2.2 Client Window

- API - Client Window

2.3 Event Driven Callbacks

- API - Callbacks
- Definition - Callback Interface Functions
- Enumeration - Callback Events

2.4 API - General

- API - Util - Image Related
- API - Util - Matcher
- API - Util - NFIQ
- API - Util - Encryption
- API - Util - PAD
- API - Util - Lock and Key
- API - General

Chapter 3

How To

3.1 Index

3.2 How To Enable Encryption

3.2.1 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 “Disable” as default.

3.2.2 Support

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)

3.2.3 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

3.2.4 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");\n
```

3. Check status

```
IBSU_GetProperty(devicehandle, ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION, szStatus);\n
```

   (a) When Encryption is enabled
   szStatus = “TRUE”

   (b) When Encryption is disabled
   szSatatus = “FALSE”

3.3 How To Use IBScanNFIQ2

3.3.1 Description

IB provides a wrapper around NFIQ2 in the form of IBScanNFIQ2 library software developed by the National Institute of Standards and Technology (NIST). NFIQ 2 differs from NFIQ 1 in that the scoring is from 1-100. A score of 100 is excellent while a score of 1 is poor. Please refer to more information at the link below:

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"

3.3.2 Support

Available in all IB scanners.
Available in SDK v2.0.x and later.
Available in Windows only.

3.3.3 Usage

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

3.3.4 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++;
}
```

3.4 How To Use Duplicate Finger

3.4.1 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.

3.4.2 Support

Available in all IB Fingerprint scanners.
Available in SDK v2.0.1 and later.
Available in all Operating Systems. (Windows, Linux, and Android)

3.4.3 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

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”

3.4.4 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);

3.5 How To Use Hand Checker

3.5.1 Description

IBScanUltimate supports hand geometry detection which identifies if the correct “shape” and placement of the fingerprint pattern on the platen is within expected norms. Hand Geometry is a fixed algorithm which requires no configuration for ease of use and deployment.
The hand geometry evaluation is designed for use on 2 and 4 finger slap captures. 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.

If it matches, "TRUE" is returned in Boolean type

3.5.2 Support

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)

3.5.3 Usage

Function : IBSU_IsValidFingerGeometry
Parameters : Refers to "1.1.3.43) IBSU_IsValidFingerGeometry"

3.5.4 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

3.6 How To Check Required SDK version

3.6.1 Description

The latest IB products have SDK version information available in the EEPROM memory device located on board the scanner. IBScanUltimate reads it during device-open, and determines if the current SDK version supports the device. If the version is appropriate, IBScanUltimate runs the device properly.
If the device is not supported, device-open will fail. In this case, IBScanUltimate returns IBSU_ERR_HIGHER_↩ SDK_REQUIRED(-214).
At this time, the user can check the minimum SDK version required by calling API function IBSU_GetRequiredSDKVersion().

※ Legacy IBScanners should work properly, although they do not have a minimum SDK Version information since IBScanUltimate has backwards compatibility.

3.6.2 Support

Available in ALL IB Fingerprint scanners.
Available in SDK v2.0.2.9 and later.
Available in all Operating Systems. (Windows, Linux, and Android)

3.6.3 Usage & Example

Function : IBSU_GetRequiredSDKVersion
Parameters : Refers to IBSU_GetRequiredSDKVersion()
1. Working case
Device has minimum SDK Version : 2.0.0
SDK Version : 3.0.0
Device will work with SDK because the required SDK v2.0.0 is lower than SDK version.

2. Not Working case
Device has minimum SDK Version : 3.5.0
SDK Version : 3.0.0
Device will not work with SDK because the required SDK v3.5.0 is higher than SDK version.

3.7 How To Use PAD Function

3.7.1 Performance: Working Together to Enhance Security ePAD Performance: Working Together to Enhance Security

As a leader in the biometrics research and development community, Integrated Biometrics is continuously improving our sensors' enhanced Presentation Attack Detection (ePAD) capabilities. Our algorithms are regularly adapted to meet new attack strategies and materials, but adversaries are quick to evolve their tactics, techniques and procedures. The need to prevent novel, and hard to detect attacks, is a key challenge within the industry. Integrated Biometrics monitors emerging threats and actively maintains the most rigorous PAD certifications. For optimum performance, please ensure you are utilizing the most current SDK. Should you identify a new attack strategy, please inform us at technical@integratedbiometrics.com as soon as possible so we can investigate.

3.7.2 Description

For Live finger detection software level, we adapted Spoof function in our SDK.

3.7.3 Support

Available in All IB Device
Available in Windows x32/x64, Linux x32/x64, Android, ARM platform
with "PAD" package.

3.7.4 Usage

1. Configuration
Call IBSU_SetProperty() with the property as below between the function call of "IBSU_OpenDevice"(or "IBSU_↩OpenDeviceEx") and "IBSU_CallbackResultImageEx" callback.
a. Enable / Disable Spoof function on SDK
Function : IBSU_SetProperty
Property : ENUM_IBSU_PROPERTY_ENABLE_SPOOF
b. Set Spoof Sensitivity Level
Function : IBSU_SetProperty
Property : ENUM_IBSU_PROPERTY_SPOOF_LEVEL

The default Sensitivity value was determined by extensive testing.
Sensitivity should only be adjusted if you experience difficulties.
Increase the Sensitivity value to decrease the probability of detecting live fingers (and increase spoof detection probability).
Decrease the Sensitivity value to decrease the probability of detecting spoofs (and increase live finger detection probability).
For security and other reasons, it is not possible for IB to provide greater details about Sensitivity, scores and

thresholds. We apologize for the inconvenience.

2. Verification
a. Check Spoof Enable/Disable
Function : IBSU_GetProperty
Property : ENUM_IBSU_PROPERTY_ENABLE_SPOOF
b. Check Spoof Sensitity Level
Function : IBSU_GetProperty
Property : ENUM_IBSU_PROPERTY_SPOOF_LEVEL

3.7.5 Example

1. Set Spoof related values
a. Enable Spoof function :
IBSU_SetProperty(devicehandle, ENUM_IBSU_PROPERTY_ENABLE_SPOOF, "TRUE");

b. Disable Spoof function :
IBSU_SetProperty(devicehandle, ENUM_IBSU_PROPERTY_ENABLE_SPOOF,"FALSE");
c. Set Sensitivity/Level of Spoof "3"
IBSU_SetProperty(devicehandle, ENUM_IBSU_PROPERTY_SPOOF_LEVEL, "3");

2. Check Spoof setting related status :
a. Spoof Enable/Disable
IBSU_GetProperty(devicehandle, ENUM_IBSU_PROPERTY_ENABLE_SPOOF, szStatus);

case1) When Spoof is enabled
szStatus = "TRUE"
case2) When Spoof is disabled
szSatatus = "FALSE"
b. Spoof Sensitivity Level
IBSU_GetProperty(devicehandle, ENUM_IBSU_PROPERTY_SPOOF_LEVEL, szStatus);

case1) When Spoof sensitivity level set 3
szStatus = "3"
case2) When Spoof sensitivity level set 5
szSatatus = "5"

3. 'IBSU_IsSpoofFingerDetected' example on "OnEvent_ResultImageEx" callback

```
void CIBScanUltimate_SampleForVCDlg::OnEvent_ResultImageEx
{
   ...
   for (int i=0; i<segmentImageArrayCount; i++)
   {
       nRc = IBSU_IsSpoofFingerDetected(deviceHandle,
           pSegmentImageArray[i],&isSpoof);
       if(nRc != IBSU_STATUS_OK)
       {
           pDlg->SetDlgItemText(IDC_EDIT_SPOOF_RESULT, _T("Error"));
           pDlg->PostMessage( WM_USER_CAPTURE_SEQ_NEXT );
           LeaveCriticalSection(&g_CriticalSection);
           return;
       }
       if (isSpoof == TRUE)
           isSpoofStr[i].Format("%s","FAKE");
       else
           isSpoofStr[i].Format("%s","LIVE");
   }
   ...
```

```
}
```

Chapter 4

Revision History for the files

4.1 IBScanUltimate.h

2015/04/07

- Added enumeration value to IBSU_ImageData. (ProcessThres)

2013/08/03

- Reformatted.

2012/04/12

- Created.

4.2 IBScanUltimateApi.h

2024/05/23 4.0.1

- IBSU_UnloadLibrary() Not supported. Return value changed to IBSU_ERR_NOT_SUPPORT(-3)
  IBSU_SetEncryptionKey() Not supported. Return value changed to IBSU_ERR_NOT_SUPPORT(-3)
  Deprecated API function of IBSU_GetImageWidth()
  Deprecated API function of IBSU_CheckWetFinger()

2022/04/13 3.9.0

- Added API function to ISO(ISO 19794, ANSI/INCITS) template create (IBSU_ConvertImageTo↩ ISOANSI)
  Re-added IBSU_GetIBSM_ResultImageInfo() to support IBSU_ConvertImageToISOANSI API

2021/12/17 3.8.0

- Removed the code for WinCE

2021/08/04 3.7.2

- Added API function to enhance NFIQ function and support new Spoof(PAD) function (IBSU_Get↩ NFIQScoreEx, IBSU_IsSpoofFingerDetected)

2020/10/06 3.7.0

- Added API function to support new locking feature (IBSU_SetCustomerKey, IBSU_GetErrorString)

2019/06/21 3.0.0

- Added API function (IBSU_SetEncryptionKey)

2019/02/19 2.1.0

- Added API function (IBSU_GetRequiredSDKVersion)

2018/04/27 2.0.1

- Added API function to improve the display speed on Embedded System (IBSU_RemoveFingerImage(), IBSU_AddFingerImage(), IBSU_IsFingerDuplicated(), IBSU_IsValidFingerGeometry())
  Deprecated the API function IBScanMater(IBSM) (IBSU_GetIBSM_ResultImageInfo())

2018/03/06 2.0.0

- Added API function to improve display speed on Embedded System (IBSU_GenerateDisplayImage())

2017/06/17 1.9.8

- Added API function to support an improved feature for CombineImage (IBSU_CheckWetFinger(), IBSU_BGetRollingInfoEx(), IBSU_GetImageWidth(), IBSU_IsWritableDirectory())

2017/04/27 1.9.7

- Added API function to support an improved feature for CombineImage (IBSU_CombineImageEx())

2015/12/11 1.9.0

- Added an API function to support Kojak devices (IBSU_GetOperableBeeper(), IBSU_SetBeeper())

2015/08/05 1.8.5

- Added an API function to combine two images into one (IBSU_CombineImage())

2015/04/07 1.8.4

- Added an API function to unload the library manually (IBSU_UnloadLibrary())

2015/03/04 1.8.3

- Reformatted to support UNICODE for WinCE
  Added an API function relelated to ClientWindow (IBSU_RedrawClientWindow())
  Bug Fixed, and added a new parameter (pitch) to WSQ functions (IBSU_WSQEncodeMem(), IBSU_WSQEncodeToFile(), IBSU_WSQDecodeMem(), IBSU_WSQDecodeFromFile())

2014/09/17 1.8.1

- Added API functions relelated to JPEG2000 and PNG (IBSU_SavePngImage(), IBSU_SaveJP2Image())

2014/07/23 1.8.0

- Added API functions relelated to WSQ (IBSU_WSQEncodeMem(), IBSU_WSQEncodeToFile(), IBSU_WSQDecodeMem(), IBSU_WSQDecodeFromFile(), IBSU_FreeMemory())

2013/10/14 1.7.0

- Added API functions to acquire an image from a device (blocking for resultEx), deregister a callback function, show (or remove) an overlay object, show (or remove) all overlay objects, add an overlay text, modify an existing overlay text, add an overlay line, modify an existing line, add an overlay quadrangle, modify an existing quadrangle, add an overlay shape, modify an overlay shape, save image to bitmap memory (IBSU_BGetImageEx(), IBSU_ReleaseCallbacks(), IBSU_ShowOverlayObject, IBSU_ShowAllOverlayObject(), IBSU_RemoveOverlayObject(), IBSU_RemoveAllOverlayObject(), IBSU_AddOverlayText(), IBSU_ModifyOverlayText(), IBSU_AddOverlayLine(), IBSU_ModifyOverlayLine(), IBSU_AddOverlayQuadrangle(), IBSU_ModifyOverlayQuadrangle(), IBSU_AddOverlayShape(), IBSU_ModifyOverlayShape(), IBSU_SaveBitmapMem())

2013/08/03 1.6.9

- Reformatted.

2013/04/05 1.6.2

- Added an API function to enable or disable trace log at run-time (IBSU_EnableTraceLog()).

2013/03/20 1.6.0

- Added an API function to support IBScanMatcher integration (IBSU_GetIBSM_ResultImageInfo(), IBSU_GetNFIQScore()).

2012/11/06 1.4.1

- Added rolling and extended open API functions (IBSU_BGetRollingInfo(), IBSU_OpenDeviceEx()).

2012/05/29 1.1.0

- Added blocking API functions (IBSU_AsyncOpenDevice(), IBSU_BGetImage(), IBSU_BGetInitProgress(), IBSU_BGetClearPlatenAtCapture()).

2012/04/06 1.0.0

- Created.

4.3 IBScanUltimateApi_defs.h

2024/03/27 4.0.0

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_DR_MODE_ZOOM_IN)

2022/05/30 3.9.1

- Removed ENUM_IBSU_PROPERTY_RESERVED_SPOOF_LEVEL_THRESHOLD

2022/04/13 3.9.0

- Added new enumerations & structures to support ISO/ANSI template integration (IBSM_TemplateVersion, IBSM_Template, IBSM_StandardFormat, IBSM_StandardFormatData)

2022/02/18 3.8.1

- Modified enumeration name changed from ENUM_IBSU_PROPERTY_VERTICAL_DIRECTION_SEGMENT to ENUM_IBSU_PROPERTY_DISABLE_SEGMENT_ROTATION ENUM_IBSU_PROPERTY_ADAPTIVE_CAPTURE_MODE property value set not permitted

2021/12/17 3.8.0

- Removed the code for WinCE

2021/07/16 3.7.2

- Added enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_VERTICAL_DIRECTION_SEGMENT, ENUM_IBSU_PROPERTY_RESERVED_SPOOF_LEVEL_THRESHOLD)

2020/10/06 3.7.0

- Modified IBSU_DeviceDesc structure for a new locking feature. Added IBSU_HashType

2020/08/21 3.5.0

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_ADAPTIVE_CAPTURE_MODE, ENUM_IBSU_PROPERTY_ENABLE_KOJAK_BEHAVIOR_2_6)

2020/04/01 3.3.0

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TRICK_CAPTURE)

2020/04/01 3.3.0

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_ROLL_METHOD, ENUM_IBSU_PROPERTY_RENEWAL_OPPOSITE_IMGAE_LEVEL, ENUM_IBSU_PROPERTY_PREVIEW_IMAGE_QUALITY_FOR_KOJAK, ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE_LEVEL, ENUM_IBSU_PROPERTY_RESERVED_ENABLE_SLIP_DETECTION, ENUM_IBSU_PROPERTY_RESERVED_SLIP_DETECTION_LEVEL)

2020/01/09 3.2.0

- Added a Spoof related enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_IS_↩ SPOOF_SUPPORTED, ENUM_IBSU_PROPERTY_ENABLE_SPOOF, ENUM_IBSU_PROPERTY_↩ SPOOF_LEVEL)

2019/10/25 3.1.1

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_FINGERPRINT_↩ SEGMENTATION_MODE)

2019/06/21 3.0.0

- Added IBSU_EncryptionMode. Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_↩ PROPERTY_VIEW_ENCRYPTION_IMAGE_MODE)

2018/04/27 2.0.1

- Deprecated enumeration (IBSM_FingerPosition)

2017/12/05 1.10.0

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_ENABLE_↩ ENCRYPTION)

2017/06/16 1.9.8

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_WET_FINGER_↩ DETECT_LEVEL_THRESHOLD, ENUM_IBSU_PROPERTY_START_POSITION_OF_ROLLING_↩ AREA, ENUM_IBSU_PROPERTY_START_ROLL_WITHOUT_LOCK, ENUM_IBSU_PROPERTY_↩ ENABLE_TOF, ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TOF_FOR_ROLL, ENUM_IBSU↩ _PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_FLAT, ENUM_IBSU_↩ PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_ROLL, ENUM_IBSU_↩ PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE)

2017/04/27 1.9.7

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_WARNING↩ _MESSAGE_INVALID_AREA, ENUM_IBSU_PROPERTY_ENABLE_WET_FINGER_DETECT, ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL)
  Added an enumeration value to IBSU_FingerQualityState (ENUM_IBSU_QUALITY_INVALID_AREA↩ _BOTTOM)

2016/09/22 1.9.4

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE, ENUM_IBSU_PROPERTY_ROLL_IMAGE_OVERRIDE)

2016/04/20 1.9.3

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_ROLLED_IMAGE_↩ WIDTH, ENUM_IBSU_PROPERTY_ROLLED_IMAGE_HEIGHT)

2016/01/21 1.9.2

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_MIN_CAPTURE_↩ TIME_IN_SUPER_DRY_MODE)

2015/12/11 1.9.0

- Added additional LED definitions for Kojak.
  Added an enumeration value to IBSU_ImageType (ENUM_IBSU_FLAT_THREE_FINGERS)
  Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_SUPER_DRY_MODE)
  Added an enumeration value to IBSU_LedType (ENUM_IBSU_LED_TYPE_FSCAN)
  Added an enumeration for beeper (IBSU_BeeperType, IBSU_BeepPattern)
  Added an enumeration value to IBSM_CaptureDeviceTypeID (IBSM_CAPTURE_DEVICE_TYPE_ID↩ _KOJAK)
  Added a new callback function (IBSU_CallbackKeyButtons)

2015/08/05 1.8.5

- Added an enumeration value to IBSU_ClientWindowPropertyId (ENUM_IBSU_WINDOW_↩ PROPERTY_KEEP_REDRAW_LAST_IMAGE)
  Added new enumerations to support Image combine (IBSU_CombineImageWhichHand)

2015/04/07 1.8.4

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_↩ ON_RELEASE, ENUM_IBSU_PROPERTY_DEVICE_INDEX, ENUM_IBSU_PROPERTY_DEVICE_ID)
  Added a reserved enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_RESERVED_↩ IMAGE_PROCESS_THRESHOLD)

2015/03/04 1.8.3

- Reformatted to support UNICODE for WinCE
  Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_CAPTURE_AREA_↩ THRESHOLD, ENUM_IBSU_PROPERTY_ENABLE_DECIMATION)
  Added an enumeration value to IBSU_ClientWindowPropertyId (ENUM_IBSU_WINDOW_↩ PROPERTY_SCALE_FACTOR_EX)
  Added a reserved feature on ENUM_IBSU_PROPERTY_RESERVED_1

2014/06/19 1.7.3

- Added an enumeration value to IBSU_ClientWindowPropertyId (ENUM_IBSU_WINDOW_↩ PROPERTY_ROLL_GUIDE_LINE_WIDTH)

2014/02/25 1.7.1

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_ROLL_MODE,ENUM↩ _IBSU_PROPERTY_ROLL_LEVEL)

2013/10/14 1.7.0

- Added new defines to support image segmentation and bitmap header. (IBSU_MAX_SEGMENT↩ _COUNT, IBSU_BMP_GRAY_HEADER_LEN, IBSU_BMP_RGB24_HEADER_LEN, IBSU_BMP_↩ RGB32_HEADER_LEN)
  Added a new structure to support image segmentation. (IBSU_SegmentPosition)
  Added new items in existing enumerations to support capture timeout, roll-fingerprint's width, draw arrow, getting the scale of the displayed image, getting the left/top margin of the display, showing invalid finger position area of top/left/right, and identifying the extended result image available callback.
  Added a new enumeration to make overlay object on client window. (IBSU_OverlayShapePattern)
  Added a new callback function (IBSU_CallbackResultImageEx)

2013/08/03 1.6.9

- Reformatted.

2013/07/31 1.6.8

- Added new defines to support Curve (TBN240) LEDs (IBSU_LED_SCAN_CURVE_RED, IBSU_LED↩ _SCAN_CURVE_GREEN, IBSU_LED_SCAN_CURVE_BLUE)

2013/03/20 1.6.0

- Added new enumerations & structures to support IBScanMatcher integration (IBSM_ImageFormat, IBSM_ImpressionType, IBSM_FingerPosition, IBSM_CaptureDeviceTechID, IBSM_CaptureDevice↩ TypeID, IBSM_CaptureDeviceVendorID, IBSM_ImageData)

2013/02/01 1.5.0

- Added an enumeration value to IBSU_ImageType (ENUM_IBSU_FLAT_FOUR_FINGERS)
  Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_RETRY_WRONG_↩ COMMUNICATION)
  Added callback function (IBSU_CallbackNotifyMessage)

2012/11/06 1.4.1

- Added an enumeration for RollingState (IBSU_RollingState).

2012/09/05 1.3.0

- Added an enumeration value to IBSU_PropertyId (ENUM_IBSU_PROPERTY_ENABLE_POWER_↩ SAVE_MODE)
  Modified IBSU_DeviceDesc structure for Android.

2012/05/29 1.1.0

- Added callback function (IBSU_CallbackAsyncOpenDevice).

2012/04/06 1.0.0

- Created.

4.4 IBScanUltimateApi_err.h

2023/02/16 3.9.3 Added error codes for KISA Matcher feature

- (IBSU_WRN_MATCHER_NO_MATCH, IBSU_WRN_MATCHER_ALREADY_REGISTERED)

2021/06/23 3.7.2

- Added error codes (IBSU_ERR_DEVICE_LOCK_INVALID_SERIAL_FORMAT, IBSU_ERR_PAD_↩ PROPERTY_DISABLED)

2020/10/06 3.7.0

- Added error codes (IBSU_ERR_DEVICE_LOCK_INVALID_BUFF, IBSU_ERR_DEVICE_LOCK_↩ INFO_EMPTY, IBSU_ERR_DEVICE_LOCK_INFO_NOT_MATCHED, IBSU_ERR_DEVICE_LOCK_↩ INVALID_CHECKSUM, IBSU_ERR_DEVICE_LOCK_INVALID_KEY, IBSU_ERR_DEVICE_LOCK_↩ LOCKED, IBSU_ERR_DEVICE_LOCK_ILLEGAL_DEVICE)

2020/04/01 3.3.0

- Added warning codes (IBSU_WRN_ROLLING_SLIP_DETECTED, IBSU_WRN_SPOOF_INIT_↩ FAILED)

2020/01/09 3.2.0

- Added warning codes (IBSU_WRN_SPOOF_DETECTED)

2018/04/27 2.0.1

- Added error codes (IBSU_ERR_DEVICE_INVALID_CALIBRATION_DATA, IBSU_ERR_DUPLICATE↩ _EXTRACTION_FAILED, IBSU_ERR_DUPLICATE_ALREADY_USED, IBSU_ERR_DUPLICATE_↩ SEGMENTATION_FAILED, IBSU_ERR_DUPLICATE_MATCHING_FAILED)

2017/06/16 1.9.8

- Added error codes (IBSU_ERR_DEVICE_NEED_CALIBRATE_TOF, IBSU_WRN_MULTIPLE_↩ FINGERS_DURING_ROLL)

2017/04/27 1.9.7

- Added warning codes (IBSU_WRN_QUALITY_INVALID_AREA, IBSU_WRN_INVALID_BRIGHTNESS↩ _FINGERS, IBSU_WRN_WET_FINGERS)

2015/04/07 1.8.4

- Added error codes (IBSU_ERR_LIBRARY_UNLOAD_FAILED)
  Added warning codes (IBSU_WRN_ALREADY_ENHANCED_IMAGE)

2014/09/17 1.8.1

- Added error codes for JPEG2000 and PNG (IBSU_ERR_NBIS_PNG_ENCODE_FAILED,IBSU_ERR↩ _NBIS_JP2_ENCODE_FAILED)

2014/07/23 1.8.0

- Added error codes for WSQ (IBSU_ERR_NBIS_WSQ_ENCODE_FAILED,IBSU_ERR_NBIS_WSQ_↩ DECODE_FAILED)

2014/02/25 1.7.1

- Added warning to check incorrect fingers/smear (IBSU_WRN_ROLLING_SHIFTED_HORIZONTALLY,IBSU↩ _WRN_ROLLING_SHIFTED_VERTICALLY)

2013/10/14 1.7.0

- Added error codes to check update firmware, invalid overlay handle (IBSU_ERR_DEVICE_NEED_↩ UPDATE_FIRMWARE,IBSU_ERR_INVALID_OVERLAY_HANDLE)
  Added warning codes to deprecate API functions and detect no finger/ incorrect fingers/smear in result image. (IBSU_WRN_API_DEPRECATED, IBSU_WRN_NO_FINGER, IBSU_WRN_INCORRECT_↩ FINGERS, IBSU_WRN_ROLLING_SMEAR)

2013/08/03 1.6.9

- Reformatted.

2013/03/20 1.6.0

- Added error and warning codes to support IBScanMatcher integration (IBSU_ERR_NBIS_NFIQ_↩ FAILED, IBSU_WRN_EMPTY_IBSM_RESULT_IMAGE)

2012/11/06 1.4.1

- Added warning codes (IBSU_WRN_ROLLING_NOT_RUNNING)

2012/09/05 1.3.0

- Added error and warning codes (IBSU_ERR_DEVICE_ENABLED_POWER_SAVE_MODE, IBSU_↩ WRN_CHANNEL_IO_SLEEP_STATUS, IBSU_WRN_BGET_IMAGE)

2012/04/06 1.0.0

- Created.

Chapter 5

Overview

Before using this manual, software engineers should first review the getting started guide included with the SDK installation.
The diagrams below show the normal flow of how to use the SDK.

5.1 Workflow - SDK

5.2 Workflow - Between SDK and Application

Chapter 6

Revision History for the API Manual

6.1 IBScanUltimate API Manual Revision History

2023.Mar WADE

- [IBScanUltimate v3.9.3]
  API Manual renewal :
  Added chm manual file
  Added HTML manual files
  Changed PDF munual file

2022.May MILTON

- Added descriptions of new functions for IBScanUltimate v3.9.0
  IBSU_ConvertImageToISOANSI()

2022.Feb MILTON

- Modified enumeration name changed from ENUM_IBSU_PROPERTY_VERTICAL_DIRECTION_↩ SEGMENT to ENUM_IBSU_PROPERTY_DISABLE_SEGMENT_ROTATION

2021.Aug MILTON

- Added descriptions of new functions for IBScanUltimate v3.7.2 IBSU_IsSpoofFingerDetected()

2020.Sep MILTON

- Added descriptions of new functions for IBScanUltimate 3.7.0 IBSU_SetCustomerKey(), IBSU_GetErrorString()
  Added error codes for new locking feature.(-215 ∼ -221)

Modified IBSU_DeviceDesc structure

2020.Aug MILTON

- Added descriptions of Capture logic related properties for IBScanUltimate v3.5.0

2020.Apr ETHAN

- Added descriptions of Roll, Kojak related properties and warning code for IBScanUltimate v3.3.0

2020.Jan MILTON

- Added descriptions of new Spoof function / properties for IBScanUltimate v3.2.0

2019.Jun MILTON

- Added descriptions of new functions for IBScanUltimate v3.0.0 IBSU_SetEncryptionKey()

2019.Jan MILTON

- Added descriptions of new functions for IBScanUltimate v2.0.2.9 IBSU_GetRequiredSDKVersion

2018.Mar WADE Added descriptions of new functions for IBScanUltimate v2.0.0.2b

- IBSU_AddFingerImage(), IBSU_RemoveFingerImage(), IBSU_IsFingerDuplicated(), IBSU_IsValidFingerGeometry()

2018.Mar YNG

- Added API function to improve dispaly speed on Embedded System (IBSU_GenerateDisplayImage)

2017.Jun GON

- Added descriptions of new functions for IBScanUltimate v1.9.6: IBSU_CheckWetFinger(), IBSU_GetImageWidth() and IBSU_IsWritableDirectory()

2017.Apr GON

- Added descriptions of new functions for IBScanUltimate v1.9.6: IBSU_CombineImageEx()

2015.Dec 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.Aug YNG

- Added descriptions of new functions for IBScanUltimate v1.8.5 IBSU_CombineImage()

2015.Apr YNG

- Added descriptions of new functions for IBScanUltimate v1.8.4: IBSU_UnloadLibrary() Added descriptions of exist callback functions

2015.Mar 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.Sep YNG

- Added descriptions of new functions for IBScanUltimate v1.8.1: IBSU_SavePngImage(), IBSU_SaveJP2Image()

2014.Jul YNG

- Added descriptions of new functions for IBScanUltimate v1.8.0: IBSU_WSQEncodeMem(), IBSU_WSQEncodeToFile(), IBSU_WSQDecodeMem(), IBSU_WSQDecodeFromFile(), IBSU_FreeMemory()

2013.Oct 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.

Chapter 7

Structures & Constants

7.1 Structure

- Structure - ImageData
- Structure - SDK Version
- Structure - Device Description
- Structure - Property of AlcorLink Device
- Structure - Coordinates of Segments
- Structure - Matcher - Image Data
- Structure - Matcher - Template Data
- Structure - Matcher - Standard Format Data

7.2 Enumeration

- Enumeration - ImageFormat
- Enumeration - Image Type
- Enumeration - Image Resolution
- Enumeration - Property ID
- Enumeration - Property ID for Client Window
- Enumeration - Finger Count State
- Enumeration - Finger Count State
- Enumeration - LE Operation Mode
- Enumeration - Platen State
- Enumeration - Callback Events
- Enumeration - LED Type
- Enumeration - Rolling State

- Enumeration - Client Window - Overlay Pattern Type
- Enumeration - Combine Two Finger Image Type
- Enumeration - Beeper Type
- Enumeration - Beeper Pattern
- Enumeration - Encryption Mode
- Enumeration - Matcher - Image format
- Enumeration - Matcher - Image Impresstion Type
- Enumeration - Matcher - Finger Position
- Enumeration - Matcher - Capture Device Tech ID
- Enumeration - Matcher - Supported Device
- Enumeration - Matcher - Vendor ID
- Enumeration - Matcher - Hash Type
- Enumeration - Matcher - Standard Format Type
- Enumeration - Matcher - Template Version

7.3 Definitions

- Definition - General
- Definition - LED
- Definition - LED for 4-Finger Scanners
- Definition - Finger Types
- Definition - Callback Interface Functions
- Definition - Error Code - General
- Definition - Error Code - Low Level I/O
- Definition - Error Code - Device Related
- Definition - Error Code - Image Capture Related
- Definition - Error Code - Client Window Related
- Definition - Error Code - NBIS Related
- Definition - Error Code - Matcher Related
- Definition - Error Code - PAD Related
- Definition - Error Code - ISO/ANSI
- Definition - Warning Code - General
- Definition - Warning Code - Smear
- Definition - Warning Code - Invalid Area

Chapter 8

Support Contact Information

Please visit Integrated Biometrics homepage http://integratedbiometrics.com to get in contact with the technical supporters and maintainers.

The staff at Integrated Biometrics is dedicated to sharing best practices in mobile fingerprint biometric technology, actively participating at industry events around the world. The cement holding our knowledgebase together is the technical know-how of our product development team and support technicians, who are committed to keeping up-to-date on the ever-changing nuances of mobile sensor technology.

"The Integrated Biometric devices are solid, provide highquality images and integrated into our software with less effort than any other device." – Lakota Software Solutions, Aaron Wilson, President/CEO

8.1 Correspondence & Address

P.O. Box 170938
spartanbug, SC 29301

121 Broadcase Drive, Level 2
Spartanburg, SC 29303
+1-864-990-3711

#703
474 Dunchon-daero
Jungwon-gu, Seongnam-si, Gyeonggi-do,
Republic of Korea
+82-31-777-2200

8.2 Global Sales Contacts

If you need to contact individually any of sales managers, there follows their contacts:

8.2.1 United States & Canada

+1-714-305-9232
info@integratedbiometrics.com

8.2.2 Brazil

+55-41-30280222
info.brazil@integratedbiometrics.com

8.2.3 Middle East

+1-864-990-3711
info.me@integratedbiometrics.com

8.2.4 Africa

+27-73-186-6480
info.africa@integratedbiometrics.com

8.2.5 Europe

+1-404-983-6217
info.brazil@integratedbiometrics.com

8.2.6 South Korea

+82-31-777-2200
info.kr@integratedbiometrics.com

8.2.7 Asia & Australia

+65-9835-3113
info.apac@integratedbiometrics.com

8.2.8 Latin America

+1-864-990-3711
info.latam@integratedbiometrics.com

Chapter 9

Module Index

9.1 Modules

Here is a list of all modules:

Chapter 10

Class Index

10.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 11

File Index

11.1 File List

Here is a list of all files with brief descriptions:

Chapter 12

Module Documentation

12.1 Enumeration - ImageFormat

Enumeration of image formats.

Enumerations

- enum IBSU_ImageFormat { IBSU_IMG_FORMAT_GRAY , IBSU_IMG_FORMAT_RGB24 , IBSU_IMG_FORMAT_RGB32 , IBSU_IMG_FORMAT_UNKNOWN }

12.1.1 Detailed Description

Enumeration of image formats.

12.1.2 Enumeration Type Documentation

12.1.2.1 IBSU_ImageFormat

```
enum IBSU_ImageFormat
```

Enumerator

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.

Definition at line 46 of file IBScanUltimate.h.

```
00047     {
00049         IBSU_IMG_FORMAT_GRAY,
00051         IBSU_IMG_FORMAT_RGB24,
00053         IBSU_IMG_FORMAT_RGB32,
00055         IBSU_IMG_FORMAT_UNKNOWN
00056     }
```

12.2 Structure - ImageData

Container for image data and metadata.

Classes

- struct IBSU_ImageData

12.2.1 Detailed Description

Container for image data and metadata.

12.3 API - Device - Open/Close

These API functions are related to opening or closing a device.

Functions

- int WINAPI IBSU_OpenDevice (const int deviceIndex, int *pHandle)

  Initializes a device, given a particular device index.
- int WINAPI IBSU_OpenDeviceEx (const int deviceIndex, LPCSTR uniformityMaskPath, const BOOL asyncOpen, int *pHandle)

  Extension of initialize device(fast mode), given by a particular device index.
- int WINAPI IBSU_AsyncOpenDevice (const int deviceIndex)

  Asynchronous Initialize device, given by a particular device index.
- int WINAPI IBSU_CloseDevice (const int handle)

  Releases a device (by device handle).
- int WINAPI IBSU_CloseAllDevice ()

  Releases all currently initialized devices (particular device handle not needed).
- int WINAPI IBSU_IsDeviceOpened (const int handle)

  Check if a particular device is opened/initialized.

12.3.1 Detailed Description

These API functions are related to opening or closing a device.

12.3.1.1 page_API_Device_Open_Close

12.3.2 Function Documentation

12.3.2.1 IBSU_AsyncOpenDevice()

```
int WINAPI IBSU_AsyncOpenDevice (
            const int deviceIndex )
```

Asynchronous Initialize device, given by a particular device index.

Parameters

in	*deviceIndex*	Zero-based device index for device to init.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

Remarks

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

12.3.2.2 IBSU_CloseAllDevice()

```
int WINAPI IBSU_CloseAllDevice ( )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.
IBSU_ERR_RESOURCE_LOCKED: A callback is still active.

Remarks

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

12.3.2.3 IBSU_CloseDevice()

```
int WINAPI IBSU_CloseDevice (
            const int handle )
```

Releases a device (by device handle).

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.
IBSU_ERR_RESOURCE_LOCKED: A callback is still active.
IBSU_ERR_DEVICE_NOT_INITIALIZED: device(s) in use are identified by index; so either device has already been released or is unknown.

12.3.2.4 IBSU_IsDeviceOpened()

```
int WINAPI IBSU_IsDeviceOpened (
            const int handle )
```

Check if a particular device is opened/initialized.

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.

Returns

IBSU_STATUS_OK, if device is ready to be used. Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.
IBSU_ERR_INVALID_PARAM_VALUE: if handle value is out of valid range. IBSU_ERR_DEVICE_NOT_INITIALIZED: device is not initialized. IBSU_ERR_DEVICE_IO: device is initialized, but there was a communication problem.

12.3.2.5 IBSU_OpenDevice()

```
int WINAPI IBSU_OpenDevice (
            const int deviceIndex,
            int * pHandle )
```

Initializes a device, given a particular device index.

Parameters

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

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

Remarks

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

12.3.2.6 IBSU_OpenDeviceEx()

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

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

Parameters

in	*deviceIndex*	Zero-based device index for device to init.
in	*uniformityMaskPath*	Uniformity mask path in your computer. If the file does not exist or path differs, the DLL makes a new file in path.
in	*ayncnOpen*	async open device(TRUE) or sync open device(FALSE).
out	*pHandle*	Function returns device handle to be used for subsequent function calls. Memory must be provided by caller.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

Remarks

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

12.4 API - Device - Infomation

These API functions are related to retrieving device information.

Functions

- int WINAPI IBSU_GetDeviceCount (int *pDeviceCount)

  Retrieve the number of connected IB USB devices. Device count function trigger for Android SDK (Reduce polling count)
- int WINAPI IBSU_GetDeviceDescription (const int deviceIndex, IBSU_DeviceDesc *pDeviceDesc)

  Retrieve detailed device information about a particular scanner by its logical index.
- int WINAPI IBSU_GetRequiredSDKVersion (const int deviceIndex, LPSTR minSDKVersion)

  Get minimum SDK version required for running.

12.4.1 Detailed Description

These API functions are related to retrieving device information.

12.4.1.1 page_API_Device_Information

12.4.2 Function Documentation

12.4.2.1 IBSU_GetDeviceCount()

```
int WINAPI IBSU_GetDeviceCount (
            int * pDeviceCount )
```

Retrieve the number of connected IB USB devices. Device count function trigger for Android SDK (Reduce polling count)

Parameters

out	*pDeviceCount*	Number of connected devices. Memory must be provided by caller.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.4.2.2 IBSU_GetDeviceDescription()

```
int WINAPI IBSU_GetDeviceDescription (
            const int deviceIndex,
            IBSU_DeviceDesc * pDeviceDesc )
```

Retrieve detailed device information about a particular scanner by its logical index.

Parameters

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

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.4.2.3 IBSU_GetRequiredSDKVersion()

```
int WINAPI IBSU_GetRequiredSDKVersion (
            const int deviceIndex,
            LPSTR minSDKVersion )
```

Get minimum SDK version required for running.

IBSU_GetRequiredSDKVersion()

Parameters

in	*deviceIndex*	Device index.
out	*deviceIndex*	Minimum SDK Version to be returned.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.5 API - Device - Property

These API functions are related to Set or Get properties of a device.

Functions

- int WINAPI IBSU_SetProperty (const int handle, const IBSU_PropertyId propertyId, LPCSTR propertyValue)
  Set a device's property value (by handle).
- int WINAPI IBSU_GetProperty (const int handle, const IBSU_PropertyId propertyId, LPSTR propertyValue)
  Retrieves a particular device's property value (by handle).

12.5.1 Detailed Description

These API functions are related to Set or Get properties of a device.

12.5.1.1 page_API_Device_Property

12.5.2 Function Documentation

12.5.2.1 IBSU_GetProperty()

```
int WINAPI IBSU_GetProperty (
            const int handle,
            const IBSU_PropertyId propertyId,
            LPSTR propertyValue )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*propertyId*	Property identifier to get value for.
out	*propertyValue*	String returning property value. Memory must be provided by a caller. This buffer should be able to hold IBSU_MAX_STR_LEN characters.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.5.2.2 IBSU_SetProperty()

```
int WINAPI IBSU_SetProperty (
            const int handle,
            const IBSU_PropertyId propertyId,
            LPCSTR propertyValue )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*propertyId*	Property identifier to set value for.
in	*propertyValue*	String containing property value.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

Remarks

Only specific property values can be set.

12.6 API - Device - Image Aquisition

These API functions are related to Image Captures.

Functions

- int WINAPI IBSU_IsCaptureAvailable (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, BOOL *pIsAvailable)

  Check if a requested capture mode is supported by the device.
- int WINAPI IBSU_BeginCaptureImage (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, const DWORD captureOptions)

  Starts image acquisition for a particular device (by handle).
- int WINAPI IBSU_CancelCaptureImage (const int handle)

  Abort image acquisition on a device that is currently scanning.
- int WINAPI IBSU_IsCaptureActive (const int handle, BOOL *pIsActive)

  Check if a particular device is actively scanning for image acquisition.
- int WINAPI IBSU_TakeResultImageManually (const int handle)

  Start image acquisition for a particular device (by handle) with image gain manually set.
- int WINAPI IBSU_GetIBSM_ResultImageInfo (const int handle, IBSM_FingerPosition fingerPosition, IBSM_ImageData *pResultImage, IBSM_ImageData *pSplitResultImage, int *pSplitResultImageCount)

  Get the result image information.
- int WINAPI IBSU_IsTouchedFinger (const int handle, int *pTouchInValue)

  Queries a particular scanner to determine if a finger is currently detected.
- int WINAPI IBSU_CheckWetFinger (const int handle, const IBSU_ImageData inImage)

  Check if the image is wet or not.
- int WINAPI IBSU_GetImageWidth (const int handle, const IBSU_ImageData inImage, int *Width_MM)

  Get the image width of input image by millimeter(mm).
- int WINAPI IBSU_ConvertImageToISOANSI (const int handle, const IBSM_ImageData *image, const int imageCount, const IBSM_ImageFormat imageFormat, const IBSM_StandardFormat STDformat, IBSM_StandardFormatData *pdata)

  Convert Image Data to Standard Format for write file. (ISO 19794-2:2005, ISO 19794-4:2005, ISO 19794-2:2011, ISO 19794-4:2011, ANSI/INCITS 378:2004, ANSI/INCITS 381:2004)

12.6.1 Detailed Description

These API functions are related to Image Captures.

12.6.1.1 page_API_Device_Image_Aquisition

12.6.2 Function Documentation

12.6.2.1 IBSU_BeginCaptureImage()

```
int WINAPI IBSU_BeginCaptureImage (
            const int handle,
            const IBSU_ImageType imageType,
            const IBSU_ImageResolution imageResolution,
            const DWORD captureOptions )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*imageType*	Image type to capture.
in	*imageResolution*	Requested capture resolution.
in	*captureOptions*	Bit coded capture options to use:

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.
IBSU_ERR_CAPTURE_STILL_RUNNING: an acquisition is currently pending and needs to be completed first.
IBSU_ERR_CAPTURE_INVALID_MODE: acquisition mode needs to be set as a prerequisite.

Remarks

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

12.6.2.2 IBSU_CancelCaptureImage()

```
int WINAPI IBSU_CancelCaptureImage (
            const int handle )
```

Abort image acquisition on a device that is currently scanning.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

IBSU_BeginCaptureImage() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.
IBSU_ERR_CAPTURE_NOT_RUNNING: no active acquisition to be aborted.

12.6.2.3 IBSU_CheckWetFinger()

```
int WINAPI IBSU_CheckWetFinger (
            const int handle,
            const IBSU_ImageData inImage )
```

Check if the image is wet or not.

IBSU_CheckWetFinger() (Deprecated)

Parameters

param [in] handle Device handle obtained by OpenDevice functions.

Parameters

inImage	Input image data which is returned from result callback.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise. See warning codes in 'IBScanUltimateApi_err'.

12.6.2.4 IBSU_ConvertImageToISOANSI()

```
int WINAPI IBSU_ConvertImageToISOANSI (
            const int handle,
```

```
            const IBSM_ImageData * image,
            const int imageCount,
            const IBSM_ImageFormat imageFormat,
            const IBSM_StandardFormat STDformat,
            IBSM_StandardFormatData * pdata )
```

Convert Image Data to Standard Format for write file. (ISO 19794-2:2005, ISO 19794-4:2005, ISO 19794-2:2011, ISO 19794-4:2011, ANSI/INCITS 378:2004, ANSI/INCITS 381:2004)

IBSU_ConvertImageToISOANSI()

Precondition

...

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*image*	Input image data for roll to slap comparison.
in	*imageCount*	Number of image.
in	*imageFormat*	Image compression format of output data.
in	*STDformat*	ISO format of output data.
out	*pdata*	Pointer to output data.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.6.2.5 IBSU_GetIBSM_ResultImageInfo()

```
int WINAPI IBSU_GetIBSM_ResultImageInfo (
            const int handle,
            IBSM_FingerPosition fingerPosition,
            IBSM_ImageData * pResultImage,
            IBSM_ImageData * pSplitResultImage,
            int * pSplitResultImageCount )
```

Get the result image information.

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*fingerPosition*	Finger position.

Parameters

out	*pResultImage*	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.
out	*pSplitResultImage*	Pointer to array of four structures that will receive 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.
out	*pSplitResultImageCount*	Pointer to variable that will receive the number of finger images split from result image.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.6.2.6 IBSU_GetImageWidth()

```
int WINAPI IBSU_GetImageWidth (
            const int handle,
            const IBSU_ImageData inImage,
            int * Width_MM )
```

Get the image width of input image by millimeter(mm).

IBSU_GetImageWidth() (Deprecated)

Parameters

param [in] handle Device handle obtained by OpenDevice functions.

Parameters

inImage	Input image data which is returned from result callback.
Width_MM	Output millimeter (width) of Input image.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise. See warning codes in 'IBScanUltimateApi_err'.

12.6.2.7 IBSU_IsCaptureActive()

```
int WINAPI IBSU_IsCaptureActive (
            const int handle,
            BOOL * pIsActive )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*Returns*	TRUE if acquisition is in progress (preview or result image acquisition). Memory must be provided by caller..

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.6.2.8 IBSU_IsCaptureAvailable()

```
int WINAPI IBSU_IsCaptureAvailable (
            const int handle,
            const IBSU_ImageType imageType,
            const IBSU_ImageResolution imageResolution,
            BOOL * pIsAvailable )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*imageType*	Image type to verify.
in	*imageResolution*	Requested capture resolution.
out	*pIsAvailable*	Returns TRUE if mode is available. Memory must be provided by caller

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.6.2.9 IBSU_IsTouchedFinger()

```
int WINAPI IBSU_IsTouchedFinger (
            const int handle,
            int * pTouchInValue )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

IBSU_BeginCaptureImage() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pTouchInValue*	TouchValue value (0 : touch off, 1 : touch on). Memory must be provided by caller

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.6.2.10 IBSU_TakeResultImageManually()

```
int WINAPI IBSU_TakeResultImageManually (
            const int handle )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

IBSU_BeginCaptureImage() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7 API - Device - General

These API functions are related to opening & closing a device.

Functions

- int WINAPI IBSU_GetContrast (const int handle, int *pContrastValue)

  Get the contrast value for a particular scanner.
- int WINAPI IBSU_SetContrast (const int handle, const int contrastValue)

  Set the contrast value for a particular scanner.
- int WINAPI IBSU_SetLEOperationMode (const int handle, const IBSU_LEOperationMode leOperationMode)

  Sets the LE operation mode (On, Off, or Auto) for a particular scanner.
- int WINAPI IBSU_GetLEOperationMode (const int handle, IBSU_LEOperationMode *pLeOperationMode)

  Get the light-emitting (LE) film operation mode for a device.
- int WINAPI IBSU_GetOperableLEDs (const int handle, IBSU_LedType *pLedType, int *pLedCount, DWORD *pOperableLEDs)

  Get operable status LED's.
- int WINAPI IBSU_GetLEDs (const int handle, DWORD *pActiveLEDs)

  Get active status LED's for a particular scanner.
- int WINAPI IBSU_SetLEDs (const int handle, const DWORD activeLEDs)

  Set active status LED's on a particular scanner.
- int WINAPI IBSU_GetOperableBeeper (const int handle, IBSU_BeeperType *pBeeperType)
- 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)

  Set the value of Beeper on a device.
- int WINAPI IBSU_BGetImage (const int handle, IBSU_ImageData *pImage, IBSU_ImageType *pImageType, IBSU_ImageData *pSplitImageArray, int *pSplitImageArrayCount, IBSU_FingerCountState *pFingerCount↩State, IBSU_FingerQualityState *pQualityArray, int *pQualityArrayCount)

  Acquire an image from a device, blocking for result. The split 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.
- 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)

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.

- int WINAPI IBSU_BGetInitProgress (const int deviceIndex, BOOL *pIsComplete, int *pHandle, int *p↩ ProgressValue)

  Get initialization progress of a device. If initialization is complete, the handle for subsequent function calls will be returned to the application.
- int WINAPI IBSU_BGetClearPlatenAtCapture (const int handle, IBSU_PlatenState *pPlatenState)

  Determine whether the platen was clear when capture was started or has since become clear.
- int WINAPI IBSU_BGetRollingInfo (const int handle, IBSU_RollingState *pRollingState, int *pRollingLineX)

  Get information about the status of the rolled print capture for a device.
- int WINAPI IBSU_BGetRollingInfoEx (const int handle, IBSU_RollingState *pRollingState, int *pRollingLineX, int *pRollingDirection, int *pRollingWidth)

  Get information about the status of the rolled print capture for a device.

12.7.1 Detailed Description

These API functions are related to opening & closing a device.

12.7.1.1 page_API_Device_General

12.7.2 Function Documentation

12.7.2.1 IBSU_BGetClearPlatenAtCapture()

```
int WINAPI IBSU_BGetClearPlatenAtCapture (
            const int handle,
            IBSU_PlatenState * pPlatenState )
```

Determine whether the platen was clear when capture was started or has since become clear.

IBSU_BGetClearPlatenAtCapture()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pPlatenState*	Pointer to variable that will receive platen state.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.2 IBSU_BGetImage()

```
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 )
```

Acquire an image from a device, blocking for result. The split 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.

IBSU_BGetImage()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*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.
out	*pImageType*	Pointer to variable that will receive image type.
out	*pSplitImageArray*	Pointer to array of four structures that will receive 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.
out	*pSplitImageArrayCount*	Pointer to variable that will receive number of finger images split from result images.
out	*pFingerCountState*	Pointer to variable that will receive finger count state.
out	*pQualityArray*	Pointer to array of four variables that will receive quality states for finger images.
out	*pQualityArrayCount*	Pointer to variable that will receive number of finger qualities.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.3 IBSU_BGetImageEx()

```
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 )
```

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.

IBSU_BGetImageEx()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pImageStatus*	Pointer to variable that will receive status from result image acquisition. See error codes in 'IBScanUltimateApi_err'.
out	*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.
out	*pImageType*	Pointer to variable that will receive image type.
out	*pDetectedFingerCount*	Pointer to variable that will receive detected finger count.
out	*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.
out	*pSegmentPositionArray*	Pointer to array of four structures that will receive position data for individual fingers split from result image.
out	*pSegmentImageArrayCount*	Pointer to variable that will receive number of finger images split from result image.
out	*pFingerCountState*	Pointer to variable that will receive finger count state.
out	*pQualityArray*	Pointer to array of four variables that will receive quality states for finger images.
out	*pQualityArrayCount*	Pointer to variable that will receive number of finger qualities.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.4 IBSU_BGetInitProgress()

```
int WINAPI IBSU_BGetInitProgress (
            const int deviceIndex,
            BOOL * pIsComplete,
            int * pHandle,
            int * pProgressValue )
```

Get initialization progress of a device. If initialization is complete, the handle for subsequent function calls will be returned to the application.

IBSU_BGetInitProgress()

Parameters

in	*deviceIndex*	Zero-based index of the scanner.
out	*pIsComplete*	Pointer to variable that will receive indicator of initialization completion.
out	*pHandle*	Pointer to variable that will receive device handle for subsequent function calls, if 'pIsComplete' receives the value TRUE.
out	*pProgressValue*	Pointer to variable that will receive initialize progress, as a percentage between 0 and 100, inclusive.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.5 IBSU_BGetRollingInfo()

```
int WINAPI IBSU_BGetRollingInfo (
            const int handle,
            IBSU_RollingState * pRollingState,
            int * pRollingLineX )
```

Get information about the status of the rolled print capture for a device.

IBSU_BGetRollingInfo()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pRollingState*	Pointer to variable that will receive rolling state.
out	*pRollingLineX*	Pointer to variable that will receive x-coordinate of current "rolling line" for display as a guide.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.6 IBSU_BGetRollingInfoEx()

```
int WINAPI IBSU_BGetRollingInfoEx (
            const int handle,
            IBSU_RollingState * pRollingState,
            int * pRollingLineX,
            int * pRollingDirection,
            int * pRollingWidth )
```

Get information about the status of the rolled print capture for a device.

IBSU_BGetRollingInfoEx()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pRollingState*	Pointer to variable that will receive rolling state.
out	*pRollingLineX*	Pointer to variable that will receive x-coordinate of current "rolling line" for display as a guide.
out	*pRollDirection*	Pointer to variable that will receive rolling direction 0 : can't determine yet 1 : left to right —> 2 : right to left <—
out	*pRollWidth*	Pointer to vairable that will receive rolling width (mm)

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.7 IBSU_GetContrast()

```
int WINAPI IBSU_GetContrast (
            const int handle,
            int * pContrastValue )
```

Get the contrast value for a particular scanner.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

IBSU_BeginCaptureImage() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pContrastValue*	Contrast value (range: 0 <= value <= IBSU_MAX_CONTRAST_VALUE). Memory must be provided by caller.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.8 IBSU_GetLEDs()

```
int WINAPI IBSU_GetLEDs (
            const int handle,
            DWORD * pActiveLEDs )
```

Get active status LED's for a particular scanner.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pActiveLEDs*	Get active LEDs. Memory must be provided by a caller.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.9 IBSU_GetLEOperationMode()

```
int WINAPI IBSU_GetLEOperationMode (
            const int handle,
            IBSU_LEOperationMode * pLeOperationMode )
```

Get the light-emitting (LE) film operation mode for a device.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pLeOperationMode*	Pointer to variable that will receive LE film operation mode.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.10 IBSU_GetOperableBeeper()

```
int WINAPI IBSU_GetOperableBeeper (
            const int handle,
            IBSU_BeeperType * pBeeperType )
```

12.7.2.11 IBSU_GetOperableLEDs()

```
int WINAPI IBSU_GetOperableLEDs (
            const int handle,
            IBSU_LedType * pLedType,
            int * pLedCount,
            DWORD * pOperableLEDs )
```

Get operable status LED's.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pLedType*	Type of LED's. Memory must be provided by caller
out	*pLedCount*	Number of LED's. Memory must be provided by caller.
out	*pOperableLEDs*	Bit pattern of operable LED's. Memory must be provided by caller.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
out	*pLedType*	Type of LED's. Memory must be provided by caller
out	*pLedCount*	Number of LED's. Memory must be provided by caller.
out	*pOperableLEDs*	Bit pattern of operable LED's. Memory must be provided by caller.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.12 IBSU_SetBeeper()

```
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 )
```

Set the value of Beeper on a device.

IBSU_SetBeeper()

Parameters

param [in] handle Device handle obtained by OpenDevice functions.

Parameters

beepPattern	Pattern of beep.
soundTone	The frequency of the sound, using a specific value. The parameter must be in the range of 0 through 2.
duration	The duration of the sound, in 25 miliseconds intervals. The parameter must be in the range of 1 through 200 at ENUM_IBSU_BEEP_PATTERN_GENERIC, in the range of 1 through 7 at ENUM_IBSU_BEEP_PATTERN_REPEAT.
reserved_1	Reserved
reserved_2	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. The parameter must be in the range of 1 through 8 reserved_2 can use the operation(start/stop of pattern repeat), 1 to start; 0 to stop

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.13 IBSU_SetContrast()

```
int WINAPI IBSU_SetContrast (
            const int handle,
            const int contrastValue )
```

Set the contrast value for a particular scanner.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

IBSU_BeginCaptureImage() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*contrastValue*	Contrast value (range: 0 <= value <= IBSU_MAX_CONTRAST_VALUE).

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.14 IBSU_SetLEDs()

```
int WINAPI IBSU_SetLEDs (
            const int handle,
            const DWORD activeLEDs )
```

Set active status LED's on a particular scanner.

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*activeLEDs*	Set active LEDs.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.7.2.15 IBSU_SetLEOperationMode()

```
int WINAPI IBSU_SetLEOperationMode (
            const int handle,
            const IBSU_LEOperationMode leOperationMode )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*leOperationMode*	LE film operation mode.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8 API - Util - Image Related

These API functions are related to Common uses for Images.

Functions

- int WINAPI IBSU_GenerateZoomOutImage (const IBSU_ImageData inImage, BYTE *outImage, const int outWidth, const int outHeight, const BYTE bkColor)

  Generate scaled version of an image.
- int WINAPI IBSU_GenerateZoomOutImageEx (const BYTE *pInImage, const int inWidth, const int inHeight, BYTE *outImage, const int outWidth, const int outHeight, const BYTE bkColor)

  Generate scaled version of an image.
- int WINAPI IBSU_SaveBitmapImage (LPCSTR filePath, const BYTE *imgBuffer, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)

  Save image to bitmap file.
- int WINAPI IBSU_SaveBitmapMem (const BYTE *inImage, const DWORD inWidth, const DWORD in↩ Height, const int inPitch, const double inResX, const double inResY, BYTE *outBitmapBuffer, const IBSU_ImageFormat outImageFormat, const DWORD outWidth, const DWORD outHeight, const BYTE bk↩ Color)

  Save image to bitmap memory.
- int WINAPI IBSU_WSQEncodeMem (const BYTE *image, const int width, const int height, const int pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText, BYTE **compressedData, int *compressedLength)

  WSQ compresses a grayscale fingerprint image.
- int WINAPI IBSU_WSQEncodeToFile (LPCSTR filePath, const BYTE *image, const int width, const int height, const int pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText)

  Save WSQ compressed grayscale fingerprint image to a specific file path.
- int WINAPI IBSU_WSQDecodeMem (const BYTE *compressedImage, const int compressedLength, BYTE **decompressedImage, int *outWidth, int *outHeight, int *outPitch, int *outBitsPerPixel, int *outPixelPer↩ Inch)

  Decompress a WSQ-encoded grayscale fingerprint image.
- int WINAPI IBSU_WSQDecodeFromFile (LPCSTR filePath, BYTE **decompressedImage, int *outWidth, int *outHeight, int *outPitch, int *outBitsPerPixel, int *outPixelPerInch)

  Decompress a WSQ-encoded grayscale fingerprint image from a specific file path.
- int WINAPI IBSU_FreeMemory (void *memblock)

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

- int WINAPI IBSU_SavePngImage (LPCSTR filePath, const BYTE *image, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)

  Save image to PNG file.
- 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)

  Save image to JPEG-2000 file.
- int WINAPI IBSU_CombineImage (const IBSU_ImageData inImage1, const IBSU_ImageData inImage2, IBSU_CombineImageWhichHand whichHand, IBSU_ImageData *outImage)

  Combine two images (2 flat fingers) into a single image (left/right hands)
- int WINAPI IBSU_CombineImageEx (const IBSU_ImageData InImage1, const IBSU_ImageData In↩ Image2, IBSU_CombineImageWhichHand WhichHand, IBSU_ImageData *OutImage, IBSU_ImageData *pSegmentImageArray, IBSU_SegmentPosition *pSegmentPositionArray, int *pSegmentImageArrayCount)

  Combine two images (2 flat fingers) into a single image (left/right hands) and return segment information.
- 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 out↩ Format, const int outQualityLevel, const BOOL outVerticalFlip)

  Generate scaled image in various formats for fast image display on canvas. This can be used instead of IBSU_GenerateZoomOutImageEx()

12.8.1 Detailed Description

These API functions are related to Common uses for Images.

12.8.1.1 page_API_Util_Image_Related

12.8.2 Function Documentation

12.8.2.1 IBSU_CombineImage()

```
int WINAPI IBSU_CombineImage (
            const IBSU_ImageData inImage1,
            const IBSU_ImageData inImage2,
            IBSU_CombineImageWhichHand whichHand,
            IBSU_ImageData * outImage )
```

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

IBSU_CombineImage()

Parameters

inImage1	Pointer to IBSU_ImageData (index and middle finger)
inImage2	Pointer to IBSU_ImageData (ring and little finger)
whichHand	Information of left or right hand
outImage	Pointer to IBSU_ImageData (1600 x 1500 fixed size image)

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.2 IBSU_CombineImageEx()

```
int WINAPI IBSU_CombineImageEx (
            const IBSU_ImageData InImage1,
            const IBSU_ImageData InImage2,
            IBSU_CombineImageWhichHand WhichHand,
            IBSU_ImageData * OutImage,
            IBSU_ImageData * pSegmentImageArray,
            IBSU_SegmentPosition * pSegmentPositionArray,
            int * pSegmentImageArrayCount )
```

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

IBSU_CombineImageEx()

Parameters

inImage1	Pointer to IBSU_ImageData (index and middle finger)
inImage2	Pointer to IBSU_ImageData (ring and little finger)
whichHand	Information of left or right hand
outImage	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

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.3 IBSU_FreeMemory()

```
int WINAPI IBSU_FreeMemory (
            void * memblock )
```

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

IBSU_FreeMemory()

Parameters

memblock	Previously allocated memory block to be freed.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.4 IBSU_GenerateDisplayImage()

```
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 )
```

Generate scaled image in various formats for fast image display on canvas. This can be used instead of IBSU_GenerateZoomOutImageEx()

IBSU_GenerateDisplayImage()

Parameters

inImage	Original grayscale image data.
inWidth	Width of input image.
in	Height Height of input image.
outImage	Pointer to buffer that will receive output image. This buffer must hold at least 'outWidth' x 'outHeight' x 'bitsPerPixel' bytes.

Parameters

outWidth	Width of output image.
outHeight	Height of output image.
outBkColor	Background color of output image.
outFormat	IBSU_ImageFormat of output image.
outQualityLevel	Image quality of output image. The parameter must be in the range of 0 through 2
outVerticalFlip	Enable/disable vertical flip of output image.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.5 IBSU_GenerateZoomOutImage()

```
int WINAPI IBSU_GenerateZoomOutImage (
            const IBSU_ImageData inImage,
            BYTE * outImage,
            const int outWidth,
            const int outHeight,
            const BYTE bkColor )
```

Generate scaled version of an image.

IBSU_GenerateZoomOutImage()

Parameters

in	*inImage*	Original image.
	[out	outImage Pointer to buffer that will receive output image. This buffer must hold at least 'outWidth' x 'outHeight' bytes.
in	*outWidth*	Width of output image.
in	*outHeight*	Height of output image.
in	*bkColor*	Background color of output image.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.6 IBSU_GenerateZoomOutImageEx()

```
int WINAPI IBSU_GenerateZoomOutImageEx (
            const BYTE * pInImage,
            const int inWidth,
            const int inHeight,
            BYTE * outImage,
            const int outWidth,
            const int outHeight,
            const BYTE bkColor )
```

Generate scaled version of an image.

IBSU_GenerateZoomOutImageEx()

Parameters

in	*inImage*	Original image data.
in	*inWidth*	Width of input image.
in	*in*	Height Height of input image.
out	*outImage*	Pointer to buffer that will receive output image. This buffer must hold at least 'outWidth' x 'outHeight' bytes.
in	*outWidth*	Width of output image.
in	*outHeight*	Height of output image.
in	*bkColor*	Background color of output image.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.7 IBSU_SaveBitmapImage()

```
int WINAPI IBSU_SaveBitmapImage (
            LPCSTR filePath,
            const BYTE * imgBuffer,
```

```
            const DWORD width,
            const DWORD height,
            const int pitch,
            const double resX,
            const double resY )
```

Save image to bitmap file.

IBSU_SaveBitmapImage()

Parameters

in	*filePath*	Path of file for output image.
in	*imgBuffer*	Pointer to image buffer.
in	*width*	Image width (in pixels).
in	*height*	Image height (in pixels).
in	*pitch*	Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
in	*resX*	Horizontal image resolution (in pixels/inch).
in	*resY*	Vertical image resolution (in pixels/inch).

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.8 IBSU_SaveBitmapMem()

```
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 )
```

Save image to bitmap memory.

IBSU_SaveBitmapMem()

Parameters

in	*inImage*	Point to image data (gray scale image).
in	*inWidth*	Image width (in pixels).
in	*inHeight*	Image height (in pixels).
in	*inPitch*	Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
in	*inResX*	Horizontal image resolution (in pixels/inch).
in	*inResY*	Vertical image resolution (in pixels/inch).
out	*outBitmapBuffer*	Pointer to output image data buffer which is a set image format and zoom-out factor. Required memory buffer size depends upon the output image format (outImageFormat): for IBSU_IMG_FORMAT_GRAY: IBSU_BMP_GRAY_HEADER_LEN + outWidth ∗ outHeight bytes for IBSU_IMG_FORMAT_RGB24: IBSU_BMP_RGB24_HEADER_LEN + 3 ∗ outWidth ∗ outHeight bytes for IBSU_IMG_FORMAT_RGB32: IBSU_BMP_RGB32_HEADER_LEN + 4 ∗ outWidth ∗ outHeight bytes
in	*outImageFormat*	Set Image color format for output image
in	*outWidth*	Width for zoom-out image
in	*outHeight*	Height for zoom-out image
in	*bkColor*	Background color for remaining area from zoom-out image

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.9 IBSU_SaveJP2Image()

```
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 )
```

Save image to JPEG-2000 file.

IBSU_SaveJP2Image()

Parameters

filePath	Path of file for output image.
imgBuffer	Pointer to image buffer.
width	Image width (in pixels).
height	Image height (in pixels).
pitch	Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
resX	Horizontal image resolution (in pixels/inch).
resY	Vertical image resolution (in pixels/inch).
fQuality	Quality level for JPEG2000, the valid range is between 0 and 100

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.10 IBSU_SavePngImage()

```
int WINAPI IBSU_SavePngImage (
            LPCSTR filePath,
            const BYTE * image,
            const DWORD width,
            const DWORD height,
            const int pitch,
            const double resX,
            const double resY )
```

Save image to PNG file.

IBSU_SavePngImage()

Parameters

filePath	Path of file for output image.
imgBuffer	Pointer to image buffer.
width	Image width (in pixels).
height	Image height (in pixels).
pitch	Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
resX	Horizontal image resolution (in pixels/inch).
resY	Vertical image resolution (in pixels/inch).

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.11 IBSU_WSQDecodeFromFile()

```
int WINAPI IBSU_WSQDecodeFromFile (
            LPCSTR filePath,
            BYTE ** decompressedImage,
            int * outWidth,
            int * outHeight,
            int * outPitch,
            int * outBitsPerPixel,
            int * outPixelPerInch )
```

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

IBSU_WSQDecodeFromFile()

Parameters

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

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.12 IBSU_WSQDecodeMem()

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

Decompress a WSQ-encoded grayscale fingerprint image.

IBSU_WSQDecodeMem()

Parameters

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

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.13 IBSU_WSQEncodeMem()

```
int WINAPI IBSU_WSQEncodeMem (
            const BYTE * image,
            const int width,
            const int height,
            const int pitch,
            const int bitsPerPixel,
            const int pixelPerInch,
            const double bitRate,
```

```
            const char * commentText,
            BYTE ** compressedData,
            int * compressedLength )
```

WSQ compresses a grayscale fingerprint image.

IBSU_WSQEncodeMem()

Parameters

image	Original image.
width	Width of original image (in pixels).
height	Height of original image (in pixels).
pitch	Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.
bitsPerPixel	Bits per pixel of original image.
pixelPerInch	Pixel per inch of original image.
bitRate	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	Comment to write compressed data.
compressedData	Pointer of image which is compressed from original image by WSQ compression. This pointer is deallocated by IBSU_FreeMemory() after using it.
compressedLength	Length of image which is compressed from original image by WSQ compression.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.8.2.14 IBSU_WSQEncodeToFile()

```
int WINAPI IBSU_WSQEncodeToFile (
            LPCSTR filePath,
            const BYTE * image,
            const int width,
            const int height,
            const int pitch,
            const int bitsPerPixel,
            const int pixelPerInch,
            const double bitRate,
            const char * commentText )
```

Save WSQ compressed grayscale fingerprint image to a specific file path.

IBSU_WSQEncodeToFile()

Parameters

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

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.9 API - Util - Matcher

This is API functions related to Matcher.

Functions

- int WINAPI IBSU_RemoveFingerImage (const int handle, const DWORD fIndex)
- int WINAPI IBSU_AddFingerImage (const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const BOOL flagForce)

  Add a finger image for the fingerprint duplicate check and roll to slap comparison. It can have only ten prints.
- int WINAPI IBSU_IsFingerDuplicated (const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const int securityLevel, DWORD ∗pMatchedPosition)

  Checks for a fingerprint duplicate from the stored prints by IBSU_AddFingerImage().
- int WINAPI IBSU_IsValidFingerGeometry (const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, BOOL ∗pValid)

  Check for hand and finger geometry whether it is correct or not.

12.9.1 Detailed Description

This is API functions related to Matcher.

12.9.1.1 page_API_Util_Matcher

/ /∗∗

IBSU_RemoveFingerImages()

Parameters

param [in] handle Device handle obtained by OpenDevice functions.

Parameters

fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING in IBScanUltimateApi_defs.h

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.9.2 Function Documentation

12.9.2.1 IBSU_AddFingerImage()

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

Add a finger image for the fingerprint duplicate check and roll to slap comparison. It can have only ten prints.

IBSU_AddFingerImage()

Parameters

param [in] handle Device handle obtained by OpenDevice functions.

Parameters

image	Input image data.
fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING in IBScanUltimateApi_defs.h

Parameters

imageType	Image type of input image.
flagForce	Indicates whether input image should be saved even if another image is already stord or not. TRUE to be stored force; FALSE to be not stored force.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.9.2.2 IBSU_IsFingerDuplicated()

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

Checks for a fingerprint duplicate from the stored prints by IBSU_AddFingerImage().

IBSU_IsFingerDuplicated()

Parameters

param [in] handle Device handle obtained by OpenDevice functions.

Parameters

image	Input image data for the fingerprint duplicate check.
fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING in IBScanUltimateApi_defs.h
imageType	Image type of input image.
securityLevel	Security level for the duplicate checks.
pMatchedPosition	Pointer to variable that will receive result of duplicate.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.9.2.3 IBSU_IsValidFingerGeometry()

```
int WINAPI IBSU_IsValidFingerGeometry (
            const int handle,
            const IBSU_ImageData image,
            const DWORD fIndex,
            const IBSU_ImageType imageType,
            BOOL * pValid )
```

Check for hand and finger geometry whether it is correct or not.

IBSU_IsValidFingerGeometry()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
	image	Input image data for roll to slap comparison.
	fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING in IBScanUltimateApi_defs.h
	imageType	Image type of input image.
	pValid	Pointer to variable that will receive whether it is valid or not. TRUE to valid; FALSE to invalid.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.9.2.4 IBSU_RemoveFingerImage()

```
int WINAPI IBSU_RemoveFingerImage (
            const int handle,
            const DWORD fIndex )
```

12.10 API - Util - NFIQ

These are API functions related to measuring NFIQ Scores.

Functions

- int WINAPI IBSU_GetNFIQScore (const int handle, const BYTE *imgBuffer, const DWORD width, const DWORD height, const BYTE bitsPerPixel, int *pScore)

  Calculate NFIQ score for an image.
- int WINAPI IBSU_GetNFIQScoreEx (const int handle, const BYTE *imgBuffer, const DWORD width, const DWORD height, const int pitch, const BYTE bitsPerPixel, int *pScore)

12.10.1 Detailed Description

These are API functions related to measuring NFIQ Scores.

12.10.1.1 page_API_Util_NFIQ

12.10.2 Function Documentation

12.10.2.1 IBSU_GetNFIQScore()

```
int WINAPI IBSU_GetNFIQScore (
            const int handle,
            const BYTE * imgBuffer,
            const DWORD width,
            const DWORD height,
            const BYTE bitsPerPixel,
            int * pScore )
```

Calculate NFIQ score for an image.

IBSU_GetNFIQScore()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*imgBuffer*	Pointer to image buffer.
in	*width*	Image width (in pixels).
in	*height*	Image height (in pixels).
in	*bitsPerPixel*	Bits per pixel.
out	*pScore*	Pointer to variable that will receive NFIQ score.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.10.2.2 IBSU_GetNFIQScoreEx()

```
int WINAPI IBSU_GetNFIQScoreEx (
            const int handle,
            const BYTE * imgBuffer,
            const DWORD width,
            const DWORD height,
            const int pitch,
            const BYTE bitsPerPixel,
            int * pScore )
```

12.11 API - Util - PAD

These are API functions related to Encryption Mode.

Functions

- int WINAPI IBSU_IsSpoofFingerDetected (const int handle, const IBSU_ImageData image, BOOL *pIsSpoof)
  Detect if the finger print is Live or Fake.

12.11.1 Detailed Description

These are API functions related to Encryption Mode.

12.11.1.1 page_API_Util_PAD

12.11.2 Function Documentation

12.11.2.1 IBSU_IsSpoofFingerDetected()

```
int WINAPI IBSU_IsSpoofFingerDetected (
            const int handle,
            const IBSU_ImageData image,
            BOOL * pIsSpoof )
```

Detect if the finger print is Live or Fake.

IBSU_IsSpoofFingerDetected()

Precondition

...

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*image*	Input image data.
out	*pIsSpoof*	Pointer to variable that will receive whether it is Spoof or Live. TRUE to Spoof; FALSE to Live.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.12 API - Util - Encryption

These are API functions related to Encryption Mode.

Functions

- int WINAPI IBSU_SetEncryptionKey (const int handle, const unsigned char *pEncyptionKey, const IBSU_EncryptionMode encMode)

  Set encryption key and mode. (Currently not supported)

12.12.1 Detailed Description

These are API functions related to Encryption Mode.

12.12.1.1 page_API_Util_Encryption

12.12.2 Function Documentation

12.12.2.1 IBSU_SetEncryptionKey()

```
int WINAPI IBSU_SetEncryptionKey (
            const int handle,
            const unsigned char * pEncyptionKey,
            const IBSU_EncryptionMode encMode )
```

Set encryption key and mode. (Currently not supported)

IBSU_SetEncryptionKey()

Precondition

...

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*pEncyptionKey*	Input data for encryption key (should be 32 bytes).
in	*encMode*	Input data for encryption mode. (random, custom)

Returns

IBSU_ERR_NOT_SUPPORTED , (this API not support currently)

12.13 API - Util - Lock and Key

These are API functions related to Encryption Mode.

Functions

- int WINAPI IBSU_SetCustomerKey (const int deviceIndex, const IBSU_HashType hashType, LPCSTR p↩ CustomerKey)

  Set CustomerKey to use locked devices, This must be perfomed on locked devices before IBSU_OpenDevice.

12.13.1 Detailed Description

These are API functions related to Encryption Mode.

12.13.1.1 page_API_Util_Lock_and_Key

12.13.2 Function Documentation

12.13.2.1 IBSU_SetCustomerKey()

```
int WINAPI IBSU_SetCustomerKey (
            const int deviceIndex,
            const IBSU_HashType hashType,
            LPCSTR pCustomerKey )
```

Set CustomerKey to use locked devices, This must be perfomed on locked devices before IBSU_OpenDevice.

IBSU_SetCustomerKey()

Precondition

The device should be locked using the IBDeviceLockWizard SW

Parameters

in	*deviceIndex*	Device index.
in	*nHashType*	Type of Hash.
in	*pCustomerKey*	Customer Key to match lock info written in the locked device.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.14 API - General

These are API functions related to Common use.

Functions

- int WINAPI IBSU_GetSDKVersion (IBSU_SdkVersion *pVerinfo)

  Gets a structure holding product and software version information (IBSU_SdkVersion).
- int WINAPI IBSU_GetSDKVersionW (IBSU_SdkVersionW *pVerinfo)
- int WINAPI IBSU_UnloadLibrary ()

  The library is unmapped from the address space explicitly, and the library is no longer valid.
- int WINAPI IBSU_IsWritableDirectory (LPCSTR dirpath, const BOOL needCreateSubFolder)

  Check whether a directory is writable.
- int WINAPI IBSU_GetErrorString (const int errorCode, LPSTR errorString)

  Returns a string description of the error code.
- int WINAPI IBSU_EnableTraceLog (const BOOL on)

  Enable or disable trace log. The trace log is enabled by default on both Windows and Android, and disabled by default on Linux.

12.14.1 Detailed Description

These are API functions related to Common use.

12.14.1.1 page_API_General

12.14.2 Function Documentation

12.14.2.1 IBSU_EnableTraceLog()

```
int WINAPI IBSU_EnableTraceLog (
            const BOOL on )
```

Enable or disable trace log. The trace log is enabled by default on both Windows and Android, and disabled by default on Linux.

Parameters

in	*on*	TRUE to enable trace log; FALSE to disable it.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.14.2.2 IBSU_GetErrorString()

```
int WINAPI IBSU_GetErrorString (
            const int errorCode,
            LPSTR errorString )
```

Returns a string description of the error code.

IBSU_GetErrorString()

Precondition

...

Parameters

in	*errorCode*	Device index.
out	*errorString*	Buffer in which value of error string description will be stored. This buffer should be able to hold IBSU_MAX_STR_LEN characters.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.14.2.3 IBSU_GetSDKVersion()

```
int WINAPI IBSU_GetSDKVersion (
            IBSU_SdkVersion * pVerinfo )
```

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

Parameters

out	*pVerinfo*	API version information. Memory must be provided by a caller.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.14.2.4 IBSU_GetSDKVersionW()

```
int WINAPI IBSU_GetSDKVersionW (
            IBSU_SdkVersionW * pVerinfo )
```

12.14.2.5 IBSU_IsWritableDirectory()

```
int WINAPI IBSU_IsWritableDirectory (
            LPCSTR dirpath,
            const BOOL needCreateSubFolder )
```

Check whether a directory is writable.

IBSU_IsWritableDirectory()

Parameters

dirpath	Directory path.
needCreateSubFolder	Check the need to create a subfolder in the directory path.

Returns

IBSU_STATUS_OK, if a directory is writable. Error code < 0, otherwise. See warning codes in 'IBScan↩ UltimateApi_err'. IBSU_ERR_CHANNEL_IO_WRITE_FAILED: Directory does not writable.

12.14.2.6 IBSU_UnloadLibrary()

```
int WINAPI IBSU_UnloadLibrary ( )
```

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

IBSU_UnloadLibrary() (Not supported)

Returns

IBSU_ERR_NOT_SUPPORTED , (this API not support currently)

12.15 API - Client Window

These are API functions related to drawing on Client Window.

Functions

- int WINAPI IBSU_CreateClientWindow (const int handle, const IBSU_HWND hWindow, const DWORD left, const DWORD top, const DWORD right, const DWORD bottom)

  Create a client window associated with a device. (Available only on Windows.)
- int WINAPI IBSU_DestroyClientWindow (const int handle, const BOOL clearExistingInfo)

  Destroy client window associated with a device. (Available only on Windows.)
- int WINAPI IBSU_GetClientWindowProperty (const int handle, const IBSU_ClientWindowPropertyId propertyId, LPSTR propertyValue)

  Get the value of a property for the client window associated with a device. For descriptions of properties and values, see definition of 'IBSU_ClientWindowPropertyId'. (Available only on Windows.)
- int WINAPI IBSU_GetClientWindowPropertyW (const int handle, const IBSU_ClientWindowPropertyId propertyId, wchar_t *propertyValue)
- int WINAPI IBSU_SetClientDisplayProperty (const int handle, const IBSU_ClientWindowPropertyId propertyId, LPCSTR propertyValue)

  Set the value of a property for the client window associated with a device. For descriptions of properties and values, see definition of 'IBSU_ClientWindowPropertyId'. (Available only on Windows.)
- int WINAPI IBSU_SetClientDisplayPropertyW (const int handle, const IBSU_ClientWindowPropertyId propertyId, const wchar_t *propertyValue)
- int WINAPI IBSU_SetClientWindowOverlayText (const int handle, const char *fontName, const int fontSize, const BOOL fontBold, const char *text, const int posX, const int posY, const DWORD textColor)

  Set the overlay text for the client window associated with a device. (Available only on Windows.) (Deprecated)
- int WINAPI IBSU_SetClientWindowOverlayTextW (const int handle, const wchar_t *fontName, const int fontSize, const BOOL fontBold, const wchar_t *text, const int posX, const int posY, const DWORD textColor)
- int WINAPI IBSU_ShowOverlayObject (const int handle, const int overlayHandle, const BOOL show)

  Show or hide an overlay object.
- int WINAPI IBSU_ShowAllOverlayObject (const int handle, const BOOL show)

  Show or hide all overlay objects.
- int WINAPI IBSU_RemoveOverlayObject (const int handle, const int overlayHandle)

  Remove an overlay object.
- int WINAPI IBSU_RemoveAllOverlayObject (const int handle)

  Remove all overlay objects.

- 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)

  Add an overlay text for display on the window.
- int WINAPI IBSU_AddOverlayTextW (const int handle, int *pOverlayHandle, const wchar_t *fontName, const int fontSize, const BOOL fontBold, const wchar_t *text, const int posX, const int posY, const DWORD text↩ Color)
- 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)

  Modify an existing overlay text for display on the window.
- int WINAPI IBSU_ModifyOverlayTextW (const int handle, const int overlayHandle, const wchar_t *fontName, const int fontSize, const BOOL fontBold, const wchar_t *text, const int posX, const int posY, const DWORD textColor)
- 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)

  Add an overlay line for display on the window.
- 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)

  Modify an existing line for display on the window.
- 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)

  Add an overlay quadrangle for display on the window.
- 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)

  Modify an existing quadrangle for display on the window.
- 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)

  Add an overlay shape for display on the window.
- 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)

  Modify an overlay shape for display on the window.
- int WINAPI IBSU_RedrawClientWindow (const int handle)

  Update the specified client window which is defined by IBSU_CreateClientWindow(). (Available only on Windows.)

12.15.1 Detailed Description

These are API functions related to drawing on Client Window.

12.15.1.1 page_API_Client_Window

12.15.2 Function Documentation

12.15.2.1 IBSU_AddOverlayLine()

```
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 )
```

Add an overlay line for display on the window.

IBSU_AddOverlayLine()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
	pOverlayHandle	Function returns overlay handle to be used for client windows function calls.
	x1	X coordinate of start point of line.
	y1	Y coordinate of start point of line.
	x2	X coordinate of end point of line.
	y2	Y coordinate of end point of line.
	lineWidth	Line width.
	lineColor	Line color.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.2 IBSU_AddOverlayQuadrangle()

```
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 )
```

Add an overlay quadrangle for display on the window.

IBSU_AddOverlayQuadrangle()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions. pOverlayHandle Function returns overlay handle to be used for client windows function calls. x1 X coordinate of 1st vertex of quadrangle. y1 Y coordinate of 1st vertex of quadrangle. x2 X coordinate of 2nd vertex of quadrangle. y2 Y coordinate of 2nd vertex of quadrangle. x3 X coordinate of 3rd vertex of quadrangle. y3 Y coordinate of 3rd vertex of quadrangle. x4 X coordinate of 4th vertex of quadrangle. y4 Y coordinate of 4th vertex of quadrangle. lineWidth Line width. lineColor Line color.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.3 IBSU_AddOverlayShape()

```
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 )
```

Add an overlay shape for display on the window.

IBSU_AddOverlayShape()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions. pOverlayHandle Function returns overlay handle to be used for client windows function calls. shapePattern Pattern of shape. If ENUM_IBSU_OVERLAY_SHAPE_ARROW, reserved_1 should be the width (in pixels) of the full base of the arrowhead, and reserved_1 should be the angle (in radians) at the arrow tip between the two sides of the arrowhead. x1 X coordinate of start point of overlay shape. y1 Y coordinate of start point of overlay shape. x2 X coordinate of end point of overlay shape. y2 Y coordinate of end point of overlay shape. lineWidth Line width. lineColor Line color. reserved_1 X reserved. reserved_2 Y reserved.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.4 IBSU_AddOverlayText()

```
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 )
```

Add an overlay text for display on the window.

IBSU_AddOverlayText()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*pOverlayHandle*	Function returns overlay handle to be used for client windows function call.
in	*fontName*	Name of font.
in	*fontSize*	Font size.
in	*fontBold*	Indicates if the font is bold.
in	*text*	Text for display on window.
in	*posX*	X coordinate of text for display on window.
in	*posY*	Y coordinate of text for display on window.
in	*textColor*	Text color.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.5 IBSU_AddOverlayTextW()

```
int WINAPI IBSU_AddOverlayTextW (
             const int handle,
             int * pOverlayHandle,
             const wchar_t * fontName,
             const int fontSize,
             const BOOL fontBold,
             const wchar_t * text,
             const int posX,
             const int posY,
             const DWORD textColor )
```

12.15.2.6 IBSU_CreateClientWindow()

```
int WINAPI IBSU_CreateClientWindow (
             const int handle,
             const IBSU_HWND hWindow,
             const DWORD left,
             const DWORD top,
             const DWORD right,
             const DWORD bottom )
```

Create a client window associated with a device. (Available only on Windows.)

IBSU_CreateClientWindow()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*hWindow*	Windows handle to draw.
in	*left*	Coordinate of left edge of rectangle.
in	*top*	Coordinate of top edge of rectangle.
in	*right*	Coordinate of right edge of rectangle.
in	*bottom*	Coordinate of bottom edge of rectangle.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.7 IBSU_DestroyClientWindow()

```
int WINAPI IBSU_DestroyClientWindow (
            const int handle,
            const BOOL clearExistingInfo )
```

Destroy client window associated with a device. (Available only on Windows.)

IBSU_DestroyClientWindow()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*clearExistingInfo*	Indicates whether the existing display information, including display properties and overlay text, will be cleared.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.8 IBSU_GetClientWindowProperty()

```
int WINAPI IBSU_GetClientWindowProperty (
            const int handle,
            const IBSU_ClientWindowPropertyId propertyId,
            LPSTR propertyValue )
```

Get the value of a property for the client window associated with a device. For descriptions of properties and values, see definition of 'IBSU_ClientWindowPropertyId'. (Available only on Windows.)

IBSU_GetClientWindowProperty()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*propertyId*	Property for which value will be set.
out	*propertyValue*	Buffer in which value of property will be stored. This buffer should be able to hold IBSU_MAX_STR_LEN characters.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.9 IBSU_GetClientWindowPropertyW()

```
int WINAPI IBSU_GetClientWindowPropertyW (
            const int handle,
            const IBSU_ClientWindowPropertyId propertyId,
            wchar_t * propertyValue )
```

12.15.2.10 IBSU_ModifyOverlayLine()

```
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 )
```

Modify an existing line for display on the window.

IBSU_ModifyOverlayLine()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions. overlayHandle Handle of overlay to modify. x1 X coordinate of start point of line. y1 Y coordinate of start point of line. x2 X coordinate of end point of line. y2 Y coordinate of end point of line. lineWidth Line width. lineColor Line color.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.11 IBSU_ModifyOverlayQuadrangle()

```
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 )
```

Modify an existing quadrangle for display on the window.

IBSU_ModifyOverlayQuadrangle()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions. overlayHandle Handle of overlay to be modified. x1 X coordinate of 1st vertex of quadrangle. y1 Y coordinate of 1st vertex of quadrangle. x2 X coordinate of 2nd vertex of quadrangle. y2 Y coordinate of 2nd vertex of quadrangle. x3 X coordinate of 3rd vertex of quadrangle. y3 Y coordinate of 3rd vertex of quadrangle. x4 X coordinate of 4th vertex of quadrangle. y4 Y coordinate of 4th vertex of quadrangle. lineWidth Line width. lineColor Line color.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.12 IBSU_ModifyOverlayShape()

```
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 )
```

Modify an overlay shape for display on the window.

IBSU_ModifyOverlayShape()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
	overlayHandle	Handle of overlay to modify.
	shapePattern	Pattern of shape. If ENUM_IBSU_OVERLAY_SHAPE_ARROW, reserved_1 should be the width (in pixels) of the full base of the arrowhead, and reserved_1 should be the angle (in radians) at the arrow tip between the two sides of the arrowhead.
	x1	X coordinate of start point of overlay shape.
	y1	Y coordinate of start point of overlay shape.
	x2	X coordinate of end point of overlay shape.
	y2	Y coordinate of end point of overlay shape.
	lineWidth	Line width.
	lineColor	Line color.
	reserved_1	X reserved.
	reserved_2	Y reserved.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.13 IBSU_ModifyOverlayText()

```
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 )
```

Modify an existing overlay text for display on the window.

IBSU_ModifyOverlayText()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*overlayHandle*	Handle of overlay to modify.
in	*fontName*	Name of font.
in	*fontSize*	Font size.
in	*fontBold*	Indicates if the font is bold.
in	*text*	Text for display on window.
in	*posX*	X coordinate of text for display on window.
in	*posY*	Y coordinate of text for display on window.
in	*textColor*	Text color.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.14 IBSU_ModifyOverlayTextW()

```
int WINAPI IBSU_ModifyOverlayTextW (
            const int handle,
            const int overlayHandle,
            const wchar_t * fontName,
            const int fontSize,
            const BOOL fontBold,
            const wchar_t * text,
            const int posX,
            const int posY,
            const DWORD textColor )
```

12.15.2.15 IBSU_RedrawClientWindow()

```
int WINAPI IBSU_RedrawClientWindow (
            const int handle )
```

Update the specified client window which is defined by IBSU_CreateClientWindow(). (Available only on Windows.)

IBSU_RedrawClientWindow()

Parameters

handle	Device handle.
flags	Bit-pattern of redraw flags. See flag codes in 'IBScanUltimateApi_def

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.16 IBSU_RemoveAllOverlayObject()

```
int WINAPI IBSU_RemoveAllOverlayObject (
            const int handle )
```

Remove all overlay objects.

IBSU_RemoveAllOverlayObject()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.17 IBSU_RemoveOverlayObject()

```
int WINAPI IBSU_RemoveOverlayObject (
            const int handle,
            const int overlayHandle )
```

Remove an overlay object.

IBSU_RemoveOverlayObject()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*overlayHandle*	Overlay handle obtained by overlay functions.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.18 IBSU_SetClientDisplayProperty()

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

Set the value of a property for the client window associated with a device. For descriptions of properties and values, see definition of 'IBSU_ClientWindowPropertyId'. (Available only on Windows.)

IBSU_SetClientDisplayProperty()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*propertyId*	Property for which value will be set.
in	*propertyValue*	Value of property to set.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.19 IBSU_SetClientDisplayPropertyW()

```
int WINAPI IBSU_SetClientDisplayPropertyW (
            const int handle,
            const IBSU_ClientWindowPropertyId propertyId,
            const wchar_t * propertyValue )
```

12.15.2.20 IBSU_SetClientWindowOverlayText()

```
int WINAPI IBSU_SetClientWindowOverlayText (
            const int handle,
            const char * fontName,
            const int fontSize,
            const BOOL fontBold,
            const char * text,
            const int posX,
            const int posY,
            const DWORD textColor )
```

Set the overlay text for the client window associated with a device. (Available only on Windows.) (Deprecated)

IBSU_SetClientWindowOverlayText()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*fontName*	Font name.
in	*fontSize*	Font size.
in	*fontBold*	Indicates if the font will be bold.
in	*text*	Text to display.
in	*posX*	X-coordinate of text.
in	*poxY*	Y-coordinate of text.
in	*textColor*	Color of text.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.21 IBSU_SetClientWindowOverlayTextW()

```
int WINAPI IBSU_SetClientWindowOverlayTextW (
              const int handle,
              const wchar_t * fontName,
              const int fontSize,
              const BOOL fontBold,
              const wchar_t * text,
              const int posX,
              const int posY,
              const DWORD textColor )
```

12.15.2.22 IBSU_ShowAllOverlayObject()

```
int WINAPI IBSU_ShowAllOverlayObject (
              const int handle,
              const BOOL show )
```

Show or hide all overlay objects.

IBSU_ShowAllOverlayObject()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*show*	If TRUE, the overlay will be shown on client window. If FALSE, the overlay will be hidden on client window.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.15.2.23 IBSU_ShowOverlayObject()

```
int WINAPI IBSU_ShowOverlayObject (
            const int handle,
            const int overlayHandle,
            const BOOL show )
```

Show or hide an overlay object.

IBSU_ShowOverlayObject()

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*overlayHandle*	Overlay handle obtained by overlay functions.
in	*show*	If TRUE, the overlay will be shown on client window. If FALSE, the overlay will be hidden on client window.

Returns

IBSU_STATUS_OK, if successful.
Error code $<$ 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.16 API - Callbacks

These are API functions related to Callback functions.

Functions

- int WINAPI IBSU_RegisterCallbacks (const int handle, const IBSU_Events event, void ∗pCallbackFunction, void ∗pContext)

  This function is used to register callback methods, utilizing event-driven programming when the state of the scanner changes.
- int WINAPI IBSU_ReleaseCallbacks (const int handle, const IBSU_Events events)

  Unregister a callback function for a particular event.

12.16.1 Detailed Description

These are API functions related to Callback functions.

12.16.1.1 page_API_Callbacks

12.16.2 Function Documentation

12.16.2.1 IBSU_RegisterCallbacks()

```
int WINAPI IBSU_RegisterCallbacks (
            const int handle,
            const IBSU_Events event,
            void * pCallbackFunction,
            void * pContext )
```

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

Precondition

IBSU_OpenDevice() or IBSU_OpenDeviceEx() or IBSU_AsyncOpenDevice() called successfully

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*event*	Event for which the callback is being registered.
in	*pCallbackFunction*	Pointer to the callback function
in	*pContext*	Pointer to user context that will be passed to callback function.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.16.2.2 IBSU_ReleaseCallbacks()

```
int WINAPI IBSU_ReleaseCallbacks (
            const int handle,
            const IBSU_Events events )
```

Unregister a callback function for a particular event.

Parameters

in	*handle*	Device handle obtained by OpenDevice functions.
in	*event*	Event for which the callback is being unregistered.

Returns

IBSU_STATUS_OK, if successful.
Error code < 0, otherwise.

See also

See error codes in 'IBScanUltimateApi_err.h'.

12.17 Definition - General

General definition.

Macros

- #define IBSU_MAX_STR_LEN 128
- #define IBSU_MIN_CONTRAST_VALUE 0
- #define IBSU_MAX_CONTRAST_VALUE 34
- #define IBSU_MAX_SEGMENT_COUNT 5
- #define IBSU_MAX_SEGMENT_QUALITY_COUNT 4
- #define IBSU_BMP_GRAY_HEADER_LEN 1078
- #define IBSU_BMP_RGB24_HEADER_LEN 54
- #define IBSU_BMP_RGB32_HEADER_LEN 54
- #define IBSU_OPTION_AUTO_CONTRAST 1
- #define IBSU_OPTION_AUTO_CAPTURE 2
- #define IBSU_OPTION_IGNORE_FINGER_COUNT 4
- #define IBSU_MAX_MINUTIAE_SIZE (255+2)

12.17.1 Detailed Description

General definition.

12.17.2 Macro Definition Documentation

12.17.2.1 IBSU_BMP_GRAY_HEADER_LEN

```
#define IBSU_BMP_GRAY_HEADER_LEN 1078
```

Required buffer length for the 8bit bitmap header.

Definition at line 235 of file IBScanUltimateApi_defs.h.

12.17.2.2 IBSU_BMP_RGB24_HEADER_LEN

```
#define IBSU_BMP_RGB24_HEADER_LEN 54
```

Required buffer length for the 24bit bitmap header.

Definition at line 237 of file IBScanUltimateApi_defs.h.

12.17.2.3 IBSU_BMP_RGB32_HEADER_LEN

```
#define IBSU_BMP_RGB32_HEADER_LEN 54
```

Required buffer length for the 32bit bitmap header.

Definition at line 239 of file IBScanUltimateApi_defs.h.

12.17.2.4 IBSU_MAX_CONTRAST_VALUE

```
#define IBSU_MAX_CONTRAST_VALUE 34
```

Maximum value of contrast.

Definition at line 228 of file IBScanUltimateApi_defs.h.

12.17.2.5 IBSU_MAX_MINUTIAE_SIZE

```
#define IBSU_MAX_MINUTIAE_SIZE (255+2)
```

Maximum value of Minutiae size. (For support ISO Template control)

Definition at line 247 of file IBScanUltimateApi_defs.h.

12.17.2.6 IBSU_MAX_SEGMENT_COUNT

```
#define IBSU_MAX_SEGMENT_COUNT 5
```

Required buffer length for segment parameters.

Definition at line 231 of file IBScanUltimateApi_defs.h.

12.17.2.7 IBSU_MAX_SEGMENT_QUALITY_COUNT

```
#define IBSU_MAX_SEGMENT_QUALITY_COUNT 4
```

Required buffer length for segment quality parameters.

Definition at line 233 of file IBScanUltimateApi_defs.h.

12.17.2.8 IBSU_MAX_STR_LEN

```
#define IBSU_MAX_STR_LEN 128
```

Required length of buffer for return string parameters.

Definition at line 222 of file IBScanUltimateApi_defs.h.

12.17.2.9 IBSU_MIN_CONTRAST_VALUE

```
#define IBSU_MIN_CONTRAST_VALUE 0
```

Minimum value of contrast.

Definition at line 225 of file IBScanUltimateApi_defs.h.

12.17.2.10 IBSU_OPTION_AUTO_CAPTURE

```
#define IBSU_OPTION_AUTO_CAPTURE 2
```

Definition at line 243 of file IBScanUltimateApi_defs.h.

12.17.2.11 IBSU_OPTION_AUTO_CONTRAST

```
#define IBSU_OPTION_AUTO_CONTRAST 1
```

Capture options. For more information, see IBSU_BeginCaptureImage().

Definition at line 242 of file IBScanUltimateApi_defs.h.

12.17.2.12 IBSU_OPTION_IGNORE_FINGER_COUNT

```
#define IBSU_OPTION_IGNORE_FINGER_COUNT 4
```

Definition at line 244 of file IBScanUltimateApi_defs.h.

12.18 Definition - LED

LED bit defines.

Macros

- #define IBSU_LED_NONE 0x00000000
- #define IBSU_LED_ALL 0xFFFFFFFF
- #define IBSU_LED_INIT_BLUE 0x00000001
- #define IBSU_LED_SCAN_GREEN 0x00000002
- #define IBSU_LED_SCAN_CURVE_RED 0x00000010
- #define IBSU_LED_SCAN_CURVE_GREEN 0x00000020
- #define IBSU_LED_SCAN_CURVE_BLUE 0x00000040

12.18.1 Detailed Description

LED bit defines.

12.18.2 Macro Definition Documentation

12.18.2.1 IBSU_LED_ALL

```
#define IBSU_LED_ALL 0xFFFFFFFF
```

All LEDs on.

Definition at line 268 of file IBScanUltimateApi_defs.h.

12.18.2.2 IBSU_LED_INIT_BLUE

```
#define IBSU_LED_INIT_BLUE 0x00000001
```

Reserved for vendor (user cannot control it).

Definition at line 270 of file IBScanUltimateApi_defs.h.

12.18.2.3 IBSU_LED_NONE

```
#define IBSU_LED_NONE 0x00000000
```

All LEDs off.

Definition at line 266 of file IBScanUltimateApi_defs.h.

12.18.2.4 IBSU_LED_SCAN_CURVE_BLUE

```
#define IBSU_LED_SCAN_CURVE_BLUE 0x00000040
```

Blue LED for Curve (TBN240).

Definition at line 278 of file IBScanUltimateApi_defs.h.

12.18.2.5 IBSU_LED_SCAN_CURVE_GREEN

```
#define IBSU_LED_SCAN_CURVE_GREEN 0x00000020
```

Definition at line 276 of file IBScanUltimateApi_defs.h.

12.18.2.6 IBSU_LED_SCAN_CURVE_RED

```
#define IBSU_LED_SCAN_CURVE_RED 0x00000010
```

Red LED for Curve (TBN240).

Definition at line 274 of file IBScanUltimateApi_defs.h.

12.18.2.7 IBSU_LED_SCAN_GREEN

```
#define IBSU_LED_SCAN_GREEN 0x00000002
```

OK key - green blink phase 1.

Definition at line 272 of file IBScanUltimateApi_defs.h.

12.19 Definition - LED for 4-Finger Scanners

Specific LED bit defines with LED type = ENUM_IBSU_LED_TYPE_FSCAN (e.g four finger scanner, Kojak).

Macros

- #define IBSU_LED_F_BLINK_GREEN 0x10000000
- #define IBSU_LED_F_BLINK_RED 0x20000000
- #define IBSU_LED_F_LEFT_LITTLE_GREEN 0x01000000
- #define IBSU_LED_F_LEFT_LITTLE_RED 0x02000000
- #define IBSU_LED_F_LEFT_RING_GREEN 0x04000000
- #define IBSU_LED_F_LEFT_RING_RED 0x08000000
- #define IBSU_LED_F_LEFT_MIDDLE_GREEN 0x00100000
- #define IBSU_LED_F_LEFT_MIDDLE_RED 0x00200000
- #define IBSU_LED_F_LEFT_INDEX_GREEN 0x00400000
- #define IBSU_LED_F_LEFT_INDEX_RED 0x00800000
- #define IBSU_LED_F_LEFT_THUMB_GREEN 0x00010000
- #define IBSU_LED_F_LEFT_THUMB_RED 0x00020000
- #define IBSU_LED_F_RIGHT_THUMB_GREEN 0x00040000
- #define IBSU_LED_F_RIGHT_THUMB_RED 0x00080000
- #define IBSU_LED_F_RIGHT_INDEX_GREEN 0x00001000
- #define IBSU_LED_F_RIGHT_INDEX_RED 0x00002000
- #define IBSU_LED_F_RIGHT_MIDDLE_GREEN 0x00004000
- #define IBSU_LED_F_RIGHT_MIDDLE_RED 0x40000000
- #define IBSU_LED_F_RIGHT_RING_GREEN 0x00000100
- #define IBSU_LED_F_RIGHT_RING_RED 0x00000200
- #define IBSU_LED_F_RIGHT_LITTLE_GREEN 0x00000400
- #define IBSU_LED_F_RIGHT_LITTLE_RED 0x00000800
- #define IBSU_LED_F_PROGRESS_ROLL 0x00000010
- #define IBSU_LED_F_PROGRESS_LEFT_HAND 0x00000020
- #define IBSU_LED_F_PROGRESS_TWO_THUMB 0x00000040
- #define IBSU_LED_F_PROGRESS_RIGHT_HAND 0x00000080

12.19.1 Detailed Description

Specific LED bit defines with LED type = ENUM_IBSU_LED_TYPE_FSCAN (e.g four finger scanner, Kojak).

12.19.2 Macro Definition Documentation

12.19.2.1 IBSU_LED_F_BLINK_GREEN

```
#define IBSU_LED_F_BLINK_GREEN 0x10000000
```

All Green LEDs blink.

Definition at line 295 of file IBScanUltimateApi_defs.h.

12.19.2.2 IBSU_LED_F_BLINK_RED

```
#define IBSU_LED_F_BLINK_RED 0x20000000
```

All Red LEDs blink.

Definition at line 297 of file IBScanUltimateApi_defs.h.

12.19.2.3 IBSU_LED_F_LEFT_INDEX_GREEN

```
#define IBSU_LED_F_LEFT_INDEX_GREEN 0x00400000
```

Green LED for left index finger.

Definition at line 311 of file IBScanUltimateApi_defs.h.

12.19.2.4 IBSU_LED_F_LEFT_INDEX_RED

```
#define IBSU_LED_F_LEFT_INDEX_RED 0x00800000
```

Red LED for left index finger.

Definition at line 313 of file IBScanUltimateApi_defs.h.

12.19.2.5 IBSU_LED_F_LEFT_LITTLE_GREEN

```
#define IBSU_LED_F_LEFT_LITTLE_GREEN 0x01000000
```

Green LED for left little finger.

Definition at line 299 of file IBScanUltimateApi_defs.h.

12.19.2.6 IBSU_LED_F_LEFT_LITTLE_RED

```
#define IBSU_LED_F_LEFT_LITTLE_RED 0x02000000
```

Red LED for left little finger.

Definition at line 301 of file IBScanUltimateApi_defs.h.

12.19.2.7 IBSU_LED_F_LEFT_MIDDLE_GREEN

```
#define IBSU_LED_F_LEFT_MIDDLE_GREEN 0x00100000
```

Green LED for left middle finger.

Definition at line 307 of file IBScanUltimateApi_defs.h.

12.19.2.8 IBSU_LED_F_LEFT_MIDDLE_RED

```
#define IBSU_LED_F_LEFT_MIDDLE_RED 0x00200000
```

Red LED for left middle finger.

Definition at line 309 of file IBScanUltimateApi_defs.h.

12.19.2.9 IBSU_LED_F_LEFT_RING_GREEN

```
#define IBSU_LED_F_LEFT_RING_GREEN 0x04000000
```

Green LED for left ring finger.

Definition at line 303 of file IBScanUltimateApi_defs.h.

12.19.2.10 IBSU_LED_F_LEFT_RING_RED

```
#define IBSU_LED_F_LEFT_RING_RED 0x08000000
```

Red LED for left ring finger.

Definition at line 305 of file IBScanUltimateApi_defs.h.

12.19.2.11 IBSU_LED_F_LEFT_THUMB_GREEN

```
#define IBSU_LED_F_LEFT_THUMB_GREEN 0x00010000
```

Green LED for left thumb finger.

Definition at line 315 of file IBScanUltimateApi_defs.h.

12.19.2.12 IBSU_LED_F_LEFT_THUMB_RED

```
#define IBSU_LED_F_LEFT_THUMB_RED 0x00020000
```

Red LED for left thumb finger.

Definition at line 317 of file IBScanUltimateApi_defs.h.

12.19.2.13 IBSU_LED_F_PROGRESS_LEFT_HAND

```
#define IBSU_LED_F_PROGRESS_LEFT_HAND 0x00000020
```

LED for the left hand indicator.

Definition at line 344 of file IBScanUltimateApi_defs.h.

12.19.2.14 IBSU_LED_F_PROGRESS_RIGHT_HAND

```
#define IBSU_LED_F_PROGRESS_RIGHT_HAND 0x00000080
```

LED for the right hand indicator.

Definition at line 348 of file IBScanUltimateApi_defs.h.

12.19.2.15 IBSU_LED_F_PROGRESS_ROLL

```
#define IBSU_LED_F_PROGRESS_ROLL 0x00000010
```

LED for the roll indicator.

Definition at line 342 of file IBScanUltimateApi_defs.h.

12.19.2.16 IBSU_LED_F_PROGRESS_TWO_THUMB

```
#define IBSU_LED_F_PROGRESS_TWO_THUMB 0x00000040
```

LED for the two thumb indicator.

Definition at line 346 of file IBScanUltimateApi_defs.h.

12.19.2.17 IBSU_LED_F_RIGHT_INDEX_GREEN

```
#define IBSU_LED_F_RIGHT_INDEX_GREEN 0x00001000
```

Green LED for right index finger.

Definition at line 323 of file IBScanUltimateApi_defs.h.

12.19.2.18 IBSU_LED_F_RIGHT_INDEX_RED

```
#define IBSU_LED_F_RIGHT_INDEX_RED 0x00002000
```

Red LED for right index finger.

Definition at line 325 of file IBScanUltimateApi_defs.h.

12.19.2.19 IBSU_LED_F_RIGHT_LITTLE_GREEN

```
#define IBSU_LED_F_RIGHT_LITTLE_GREEN 0x00000400
```

Green LED for right little finger.

Definition at line 338 of file IBScanUltimateApi_defs.h.

12.19.2.20 IBSU_LED_F_RIGHT_LITTLE_RED

```
#define IBSU_LED_F_RIGHT_LITTLE_RED 0x00000800
```

Red LED for right little finger.

Definition at line 340 of file IBScanUltimateApi_defs.h.

12.19.2.21 IBSU_LED_F_RIGHT_MIDDLE_GREEN

```
#define IBSU_LED_F_RIGHT_MIDDLE_GREEN 0x00004000
```

Green LED for right middle finger.

Definition at line 327 of file IBScanUltimateApi_defs.h.

12.19.2.22 IBSU_LED_F_RIGHT_MIDDLE_RED

```
#define IBSU_LED_F_RIGHT_MIDDLE_RED 0x40000000
```

0x00008000 cannot be used at VB6 (Negative value) Red LED for right middle finger. Red LED for right middle finger.

Definition at line 332 of file IBScanUltimateApi_defs.h.

12.19.2.23 IBSU_LED_F_RIGHT_RING_GREEN

```
#define IBSU_LED_F_RIGHT_RING_GREEN 0x00000100
```

Green LED for right ring finger.

Definition at line 334 of file IBScanUltimateApi_defs.h.

12.19.2.24 IBSU_LED_F_RIGHT_RING_RED

```
#define IBSU_LED_F_RIGHT_RING_RED 0x00000200
```

Red LED for right ring finger.

Definition at line 336 of file IBScanUltimateApi_defs.h.

12.19.2.25 IBSU_LED_F_RIGHT_THUMB_GREEN

```
#define IBSU_LED_F_RIGHT_THUMB_GREEN 0x00040000
```

Green LED for right thumb finger.

Definition at line 319 of file IBScanUltimateApi_defs.h.

12.19.2.26 IBSU_LED_F_RIGHT_THUMB_RED

```
#define IBSU_LED_F_RIGHT_THUMB_RED 0x00080000
```

Red LED for right thumb finger.

Definition at line 321 of file IBScanUltimateApi_defs.h.

12.20 Definition - Finger Types

Bit-pattern of finger index for IBSU_RemoveFingerImage, IBSU_AddFingerImage, IBSU_IsFingerDuplicated and IBSU_IsValidFingerGeometry.

Macros

- #define IBSU_FINGER_NONE 0x00000000
- #define IBSU_FINGER_LEFT_LITTLE 0x00000001
- #define IBSU_FINGER_LEFT_RING 0x00000002
- #define IBSU_FINGER_LEFT_MIDDLE 0x00000004
- #define IBSU_FINGER_LEFT_INDEX 0x00000008
- #define IBSU_FINGER_LEFT_THUMB 0x00000010
- #define IBSU_FINGER_RIGHT_THUMB 0x00000020
- #define IBSU_FINGER_RIGHT_INDEX 0x00000040
- #define IBSU_FINGER_RIGHT_MIDDLE 0x00000080
- #define IBSU_FINGER_RIGHT_RING 0x00000100
- #define IBSU_FINGER_RIGHT_LITTLE 0x00000200
- #define IBSU_FINGER_LEFT_HAND (IBSU_FINGER_LEFT_INDEX | IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_RING | IBSU_FINGER_LEFT_LITTLE)
- #define IBSU_FINGER_RIGHT_HAND (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE | IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)
- #define IBSU_FINGER_BOTH_THUMBS (IBSU_FINGER_RIGHT_THUMB | IBSU_FINGER_LEFT_THUMB)
- #define IBSU_FINGER_ALL (IBSU_FINGER_LEFT_HAND | IBSU_FINGER_RIGHT_HAND | IBSU_FINGER_BOTH_THUMBS
- #define IBSU_FINGER_LEFT_LITTLE_RING (IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING)
- #define IBSU_FINGER_LEFT_MIDDLE_INDEX (IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_INDEX)
- #define IBSU_FINGER_RIGHT_INDEX_MIDDLE (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE)
- #define IBSU_FINGER_RIGHT_RING_LITTLE (IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)

12.20.1 Detailed Description

Bit-pattern of finger index for IBSU_RemoveFingerImage, IBSU_AddFingerImage, IBSU_IsFingerDuplicated and IBSU_IsValidFingerGeometry.

12.20.2 Macro Definition Documentation

12.20.2.1 IBSU_FINGER_ALL

```
#define IBSU_FINGER_ALL (IBSU_FINGER_LEFT_HAND | IBSU_FINGER_RIGHT_HAND | IBSU_FINGER_BOTH_THUMBS)
```

Definition at line 378 of file IBScanUltimateApi_defs.h.

12.20.2.2 IBSU_FINGER_BOTH_THUMBS

```
#define IBSU_FINGER_BOTH_THUMBS (IBSU_FINGER_RIGHT_THUMB | IBSU_FINGER_LEFT_THUMB)
```

Definition at line 377 of file IBScanUltimateApi_defs.h.

12.20.2.3 IBSU_FINGER_LEFT_HAND

```
#define IBSU_FINGER_LEFT_HAND (IBSU_FINGER_LEFT_INDEX | IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_RING
| IBSU_FINGER_LEFT_LITTLE)
```

Definition at line 375 of file IBScanUltimateApi_defs.h.

12.20.2.4 IBSU_FINGER_LEFT_INDEX

```
#define IBSU_FINGER_LEFT_INDEX 0x00000008
```

Definition at line 368 of file IBScanUltimateApi_defs.h.

12.20.2.5 IBSU_FINGER_LEFT_LITTLE

```
#define IBSU_FINGER_LEFT_LITTLE 0x00000001
```

Definition at line 365 of file IBScanUltimateApi_defs.h.

12.20.2.6 IBSU_FINGER_LEFT_LITTLE_RING

```
#define IBSU_FINGER_LEFT_LITTLE_RING (IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING)
```

Definition at line 379 of file IBScanUltimateApi_defs.h.

12.20.2.7 IBSU_FINGER_LEFT_MIDDLE

```
#define IBSU_FINGER_LEFT_MIDDLE 0x00000004
```

Definition at line 367 of file IBScanUltimateApi_defs.h.

12.20.2.8 IBSU_FINGER_LEFT_MIDDLE_INDEX

```
#define IBSU_FINGER_LEFT_MIDDLE_INDEX (IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_INDEX)
```

Definition at line 380 of file IBScanUltimateApi_defs.h.

12.20.2.9 IBSU_FINGER_LEFT_RING

```
#define IBSU_FINGER_LEFT_RING 0x00000002
```

Definition at line 366 of file IBScanUltimateApi_defs.h.

12.20.2.10 IBSU_FINGER_LEFT_THUMB

```
#define IBSU_FINGER_LEFT_THUMB 0x00000010
```

Definition at line 369 of file IBScanUltimateApi_defs.h.

12.20.2.11 IBSU_FINGER_NONE

```
#define IBSU_FINGER_NONE 0x00000000
```

Definition at line 364 of file IBScanUltimateApi_defs.h.

12.20.2.12 IBSU_FINGER_RIGHT_HAND

```
#define IBSU_FINGER_RIGHT_HAND (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE | IBSU_FINGER_RIGHT_RING
| IBSU_FINGER_RIGHT_LITTLE)
```

Definition at line 376 of file IBScanUltimateApi_defs.h.

12.20.2.13 IBSU_FINGER_RIGHT_INDEX

```
#define IBSU_FINGER_RIGHT_INDEX 0x00000040
```

Definition at line 371 of file IBScanUltimateApi_defs.h.

12.20.2.14 IBSU_FINGER_RIGHT_INDEX_MIDDLE

```
#define IBSU_FINGER_RIGHT_INDEX_MIDDLE (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE)
```

Definition at line 381 of file IBScanUltimateApi_defs.h.

12.20.2.15 IBSU_FINGER_RIGHT_LITTLE

```
#define IBSU_FINGER_RIGHT_LITTLE 0x00000200
```

Definition at line 374 of file IBScanUltimateApi_defs.h.

12.20.2.16 IBSU_FINGER_RIGHT_MIDDLE

```
#define IBSU_FINGER_RIGHT_MIDDLE 0x00000080
```

Definition at line 372 of file IBScanUltimateApi_defs.h.

12.20.2.17 IBSU_FINGER_RIGHT_RING

```
#define IBSU_FINGER_RIGHT_RING 0x00000100
```

Definition at line 373 of file IBScanUltimateApi_defs.h.

12.20.2.18 IBSU_FINGER_RIGHT_RING_LITTLE

```
#define IBSU_FINGER_RIGHT_RING_LITTLE (IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)
```

Definition at line 382 of file IBScanUltimateApi_defs.h.

12.20.2.19 IBSU_FINGER_RIGHT_THUMB

```
#define IBSU_FINGER_RIGHT_THUMB 0x00000020
```

Definition at line 370 of file IBScanUltimateApi_defs.h.

12.21 Structure - SDK Version

Container to hold SDK version.

Classes

- struct IBSU_SdkVersion

12.21.1 Detailed Description

Container to hold SDK version.

12.22 Structure - Device Description

Basic device description.

Classes

- struct IBSU_DeviceDesc

12.22.1 Detailed Description

Basic device description.

12.23 Structure - Property of AlcorLink Device

Container for Properteis of Alcorlink device.

Classes

- struct Property_AlcorLink

Macros

- #define PARALLEL_MAX_STR_LEN (32)

Typedefs

- typedef struct Property_AlcorLink * pProperty_AlcorLink

12.23.1 Detailed Description

Container for Properteis of Alcorlink device.

12.23.2 Macro Definition Documentation

12.23.2.1 PARALLEL_MAX_STR_LEN

```
#define PARALLEL_MAX_STR_LEN (32)
```

Definition at line 508 of file IBScanUltimateApi_defs.h.

12.23.3 Typedef Documentation

12.23.3.1 pProperty_AlcorLink

```
typedef struct Property_AlcorLink * pProperty_AlcorLink
```

12.24 Structure - Coordinates of Segments

Container to hold segment information.

Classes

- struct IBSU_SegmentPosition

12.24.1 Detailed Description

Container to hold segment information.

12.25 Enumeration - Image Type

Enumeration of image types. The image type is passed as a parameter to API functions IBSU_BeginCaptureImage() and IBSU_IsCaptureAvailable().

Enumerations

- enum IBSU_ImageType {
  ENUM_IBSU_TYPE_NONE , ENUM_IBSU_ROLL_SINGLE_FINGER , ENUM_IBSU_FLAT_SINGLE_FINGER
  , ENUM_IBSU_FLAT_TWO_FINGERS ,
  ENUM_IBSU_FLAT_FOUR_FINGERS , ENUM_IBSU_FLAT_THREE_FINGERS , ENUM_IBSU_FLAT_SINGLE_WRITERS_PA
  , ENUM_IBSU_FLAT_SINGLE_UPPER_PALM ,
  ENUM_IBSU_FLAT_SINGLE_LOWER_PALM }

12.25.1 Detailed Description

Enumeration of image types. The image type is passed as a parameter to API functions IBSU_BeginCaptureImage() and IBSU_IsCaptureAvailable().

12.25.2 Enumeration Type Documentation

12.25.2.1 IBSU_ImageType

```
enum IBSU_ImageType
```

Enumerator

ENUM_IBSU_TYPE_NONE	Unspecified type.
ENUM_IBSU_ROLL_SINGLE_FINGER	One-finger rolled fingerprint.
ENUM_IBSU_FLAT_SINGLE_FINGER	One-finger flat fingerprint.
ENUM_IBSU_FLAT_TWO_FINGERS	Two-finger flat fingerprint.
ENUM_IBSU_FLAT_FOUR_FINGERS	Four-finger flat fingerprint.
ENUM_IBSU_FLAT_THREE_FINGERS	Three-finger flat fingerprint.
ENUM_IBSU_FLAT_SINGLE_WRITERS_PALM	Palm flat writer's palm.
ENUM_IBSU_FLAT_SINGLE_UPPER_PALM	Palm flat upper plam.
ENUM_IBSU_FLAT_SINGLE_LOWER_PALM	Palm flat lower palm.

Definition at line 583 of file IBScanUltimateApi_defs.h.

```
00584 {
00586     ENUM_IBSU_TYPE_NONE,
00587
00589     ENUM_IBSU_ROLL_SINGLE_FINGER,
00590
00592     ENUM_IBSU_FLAT_SINGLE_FINGER,
00593
00595     ENUM_IBSU_FLAT_TWO_FINGERS,
00596
00598     ENUM_IBSU_FLAT_FOUR_FINGERS,
00599
00601     ENUM_IBSU_FLAT_THREE_FINGERS,
00602
00604     ENUM_IBSU_FLAT_SINGLE_WRITERS_PALM,
00605
00607     ENUM_IBSU_FLAT_SINGLE_UPPER_PALM,
00608
00610     ENUM_IBSU_FLAT_SINGLE_LOWER_PALM
00611 }
```

12.26 Enumeration - Image Resolution

Enumeration of image resolutions. The image resolution is passed as a parameter to API functions IBSU_BeginCaptureImage() and IBSU_IsCaptureAvailable().

Enumerations

- enum IBSU_ImageResolution { ENUM_IBSU_IMAGE_RESOLUTION_500 = 500 , ENUM_IBSU_IMAGE_RESOLUTION_1000 = 1000 }

12.26.1 Detailed Description

Enumeration of image resolutions. The image resolution is passed as a parameter to API functions IBSU_BeginCaptureImage() and IBSU_IsCaptureAvailable().

12.26.2 Enumeration Type Documentation

12.26.2.1 IBSU_ImageResolution

```
enum IBSU_ImageResolution
```

Enumerator

ENUM_IBSU_IMAGE_RESOLUTION_500	500 pixels/inch.
ENUM_IBSU_IMAGE_RESOLUTION_1000	1000 pixels/inch.

Definition at line 629 of file IBScanUltimateApi_defs.h.

```
00630 {
00632     ENUM_IBSU_IMAGE_RESOLUTION_500 = 500,
00633
00635     ENUM_IBSU_IMAGE_RESOLUTION_1000 = 1000
00636 }
```

12.27 Enumeration - Property ID

Enumeration of property IDs. Properties can be gotten with IBSU_GetProperty(); some properties can also be set with IBSU_SetProperty().

Enumerations

- enum IBSU_PropertyId { ENUM_IBSU_PROPERTY_PRODUCT_ID , ENUM_IBSU_PROPERTY_SERIAL_NUMBER , ENUM_IBSU_PROPERTY_VEN , ENUM_IBSU_PROPERTY_IBIA_VENDOR_ID , ENUM_IBSU_PROPERTY_IBIA_VERSION , ENUM_IBSU_PROPERTY_IBIA_DEVICE_ID , ENUM_IBSU_PROPERTY_FIRM

, ENUM_IBSU_PROPERTY_REVISION ,
ENUM_IBSU_PROPERTY_PRODUCTION_DATE , ENUM_IBSU_PROPERTY_SERVICE_DATE , ENUM_IBSU_PROPERTY_
, ENUM_IBSU_PROPERTY_IMAGE_HEIGHT ,
ENUM_IBSU_PROPERTY_IGNORE_FINGER_TIME , ENUM_IBSU_PROPERTY_RECOMMENDED_LEVEL
, ENUM_IBSU_PROPERTY_POLLINGTIME_TO_BGETIMAGE , ENUM_IBSU_PROPERTY_ENABLE_POWER_SAVE_MODE
,
ENUM_IBSU_PROPERTY_RETRY_WRONG_COMMUNICATION , ENUM_IBSU_PROPERTY_CAPTURE_TIMEOUT
, ENUM_IBSU_PROPERTY_ROLL_MIN_WIDTH , ENUM_IBSU_PROPERTY_ROLL_MODE ,
ENUM_IBSU_PROPERTY_ROLL_LEVEL , ENUM_IBSU_PROPERTY_CAPTURE_AREA_THRESHOLD ,
ENUM_IBSU_PROPERTY_ENABLE_DECIMATION , ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_ON_RELEASE
,
ENUM_IBSU_PROPERTY_DEVICE_INDEX , ENUM_IBSU_PROPERTY_DEVICE_ID , ENUM_IBSU_PROPERTY_SUPER_D
, ENUM_IBSU_PROPERTY_MIN_CAPTURE_TIME_IN_SUPER_DRY_MODE ,
ENUM_IBSU_PROPERTY_ROLLED_IMAGE_WIDTH , ENUM_IBSU_PROPERTY_ROLLED_IMAGE_HEIGHT
, ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE , ENUM_IBSU_PROPERTY_ROLL_IMAGE_OVERRIDE
,
ENUM_IBSU_PROPERTY_WARNING_MESSAGE_INVALID_AREA , ENUM_IBSU_PROPERTY_ENABLE_WET_FINGER_D
, ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL , ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL_
,
ENUM_IBSU_PROPERTY_START_POSITION_OF_ROLLING_AREA , ENUM_IBSU_PROPERTY_START_ROLL_WITHOUT_
, ENUM_IBSU_PROPERTY_ENABLE_TOF , ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION ,
ENUM_IBSU_PROPERTY_IS_SPOOF_SUPPORTED , ENUM_IBSU_PROPERTY_ENABLE_SPOOF ,
ENUM_IBSU_PROPERTY_SPOOF_LEVEL , ENUM_IBSU_PROPERTY_VIEW_ENCRYPTION_IMAGE_MODE
,
ENUM_IBSU_PROPERTY_FINGERPRINT_SEGMENTATION_MODE , ENUM_IBSU_PROPERTY_ROLL_METHOD
, ENUM_IBSU_PROPERTY_RENEWAL_OPPOSITE_IMGAE_LEVEL , ENUM_IBSU_PROPERTY_PREVIEW_IMAGE_QUAL
,
ENUM_IBSU_PROPERTY_ADAPTIVE_CAPTURE_MODE , ENUM_IBSU_PROPERTY_ENABLE_KOJAK_BEHAVIOR_2_6
, ENUM_IBSU_PROPERTY_DISABLE_SEGMENT_ROTATION , ENUM_IBSU_PROPERTY_DR_MODE_ZOOM_IN
,
ENUM_IBSU_PROPERTY_RESERVED_1 = 200 , ENUM_IBSU_PROPERTY_RESERVED_2 , ENUM_IBSU_PROPERTY_RE
, ENUM_IBSU_PROPERTY_RESERVED_IMAGE_PROCESS_THRESHOLD = 400 ,
ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TOF_FOR_ROLL , ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_B
, ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_ROLL ,
ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE ,
ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE_LEVEL , ENUM_IBSU_PROPERTY_RESERVED_E
, ENUM_IBSU_PROPERTY_RESERVED_SLIP_DETECTION_LEVEL , ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TF
,
ENUM_IBSU_PROPERTY_RESERVED_ENABLE_CBP_MODE , ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_WI
, ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_HEIGHT , ENUM_IBSU_PROPERTY_RESERVED_TFT_NOISE_RE
,
ENUM_IBSU_PROPERTY_RESERVED_LINE_BLACK_FILL , ENUM_IBSU_PROPERTY_RESERVED_RECALCILATE_BRIG
, ENUM_IBSU_PROPERTY_RESERVED_LINE_RESTORE , ENUM_IBSU_PROPERTY_RESERVED_SW_UNIFORMITY
,
ENUM_IBSU_PROPERTY_RESERVED_SET_ROLL_TEST_MODE }

12.27.1 Detailed Description

Enumeration of property IDs. Properties can be gotten with IBSU_GetProperty(); some properties can also be set with IBSU_SetProperty().

12.27.2 Enumeration Type Documentation

12.27.2.1 IBSU_PropertyId

```
enum IBSU_PropertyId
```

Enumerator

ENUM_IBSU_PROPERTY_PRODUCT_ID	Product name string (e.g., "Watson"). [Get only.]
ENUM_IBSU_PROPERTY_SERIAL_NUMBER	Serial number string. [Get only.]
ENUM_IBSU_PROPERTY_VENDOR_ID	Device manufacturer identifier. [Get only.]
ENUM_IBSU_PROPERTY_IBIA_VENDOR_ID	IBIA vendor ID. [Get only.]
ENUM_IBSU_PROPERTY_IBIA_VERSION	IBIA version information. [Get only.]
ENUM_IBSU_PROPERTY_IBIA_DEVICE_ID	IBIA device ID. [Get only.]
ENUM_IBSU_PROPERTY_FIRMWARE	Firmware version string. [Get only.]
ENUM_IBSU_PROPERTY_REVISION	Device revision string. [Get only.]
ENUM_IBSU_PROPERTY_PRODUCTION_DATE	Production date string. [Get only.]
ENUM_IBSU_PROPERTY_SERVICE_DATE	Last service date string. [Get only.]
ENUM_IBSU_PROPERTY_IMAGE_WIDTH	Image width value. [Get only.]
ENUM_IBSU_PROPERTY_IMAGE_HEIGHT	Image height value. [Get only.]
ENUM_IBSU_PROPERTY_IGNORE_FINGER_TIME	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_RECOMMENDED_↩ LEVEL	Auto contrast level value. [Get and set.]
ENUM_IBSU_PROPERTY_POLLINGTIME_TO_↩ BGETIMAGE	Polling time for IBSU_BGetImage(). [Get only.]
ENUM_IBSU_PROPERTY_ENABLE_POWER_↩ SAVE_MODE	Enable power save mode (TRUE to enable or FALSE to disable). [Get and set.]
ENUM_IBSU_PROPERTY_RETRY_WRONG_↩ COMMUNICATION	Retry count for communication error. The valid range is between 0 and 120, inclusive, with the default of 6. [Get and set.]
ENUM_IBSU_PROPERTY_CAPTURE_TIMEOUT	The maximum wait time for image capture, in seconds. Must use IBSU_CallbackResultImageEx instead of IBSU_CallbackResultImage. If -1, the timeout is infinite. Otherwise, the valid range is between 10- and 3600-seconds, inclusive. The default is -1. [Get and set.]
ENUM_IBSU_PROPERTY_ROLL_MIN_WIDTH	Minimum distance of rolled fingerprint, in millimeters. The valid range is between 10- and 30-mm, inclusive. The default is 15-mm. [Get and set.]
ENUM_IBSU_PROPERTY_ROLL_MODE	Roll mode. The valid range is between 0 $\sim$ 1. The default is 1. [Get and set.] 0 : Do not use smear 1 : use notice
ENUM_IBSU_PROPERTY_ROLL_LEVEL	Roll level. The valid range is between 0 $\sim$ 2. The default is 1. [Get and set.] 0 : low level 1 : medium level 2 : high level
ENUM_IBSU_PROPERTY_CAPTURE_AREA_↩ THRESHOLD	The area threshold for image capture for flat fingers and 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_ENABLE_DECIMATION	Enable decimation mode (TRUE to enable or FALSE to disable). Some devices (or firmware version) do not support this feature. [Get and set.]

Enumerator

ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_↩ ON_RELEASE	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 the user releases fingers from the sensor. FALSE : The result callback will be called when the quality of the finger is good
ENUM_IBSU_PROPERTY_DEVICE_INDEX	The device index. [Get only.]
ENUM_IBSU_PROPERTY_DEVICE_ID	The device ID which shares information with the UsbDevice class of Android. [Get only.]
ENUM_IBSU_PROPERTY_SUPER_DRY_MODE	This can be used for dry finger Some devices (or firmware version) do not support this feature. The default is FALSE. [Get and set.] TRUE : Enable dry mode. FALSE : Disable dry mode
ENUM_IBSU_PROPERTY_MIN_CAPTURE_TIME↩ _IN_SUPER_DRY_MODE	There is a minimum capture time when dry mode is enabled with the property ENUM_IBSU_PROPERTY_SUPER_DRY_MODE Some devices (or firmware version) do 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_ROLLED_IMAGE_↩ WIDTH	Rolled image width value. [Get only.]
ENUM_IBSU_PROPERTY_ROLLED_IMAGE_↩ HEIGHT	Rolled image height value. [Get only.]
ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE	Enable the drawing for preview image (TRUE to enable or FALSE to disable). The default is TRUE. [Get and set.]
ENUM_IBSU_PROPERTY_ROLL_IMAGE_↩ OVERRIDE	Enable to override roll image (TRUE to enable or FALSE to disable). The default is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_WARNING_↩ MESSAGE_INVALID_AREA	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_ENABLE_WET_↩ FINGER_DETECT	Enable wet detect function. The default is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_WET_FINGER_↩ DETECT_LEVEL	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_THRESHOLD	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_START_POSITION_↩ OF_ROLLING_AREA	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_ROLL_↩ WITHOUT_LOCK	Enable rolling without lock. The default is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_ENABLE_TOF	Enable TOF function. The default is set depending on the devices. [Get and set.]
ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION	Enable Encryption for capture images The default is FALSE. [Get and set.]

Enumerator

ENUM_IBSU_PROPERTY_IS_SPOOF_↩ SUPPORTED	Check if the device supprorts spoof analysis functionality [Get only.]
ENUM_IBSU_PROPERTY_ENABLE_SPOOF	Enable spoof function The default is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_SPOOF_LEVEL	Change spoof Level/Sensitivity The valid range is between 1 and 5. The default is 3. [Get and set.] The default sensitivity of 3 provides the best results through a range of test environments. The default value of 3 should be used as the baseline sensitivity when tuning application performance. An INCREASE in spoof level sensitivity ($>$ 3) will increase the number of false rejections for live fingers while decreasing the liklihood of false acceptance of spoof prints. A DECREASE in spoof level sensitivity ($<$ 3) will reduce false rejections but increases the liklihood of false acceptance of spoof prints. Spoof sensitivity is the scale for algorithm thresholds that determines the liklihood that a fingerprint is fake, or a spoof. The feature is designed so that the algorithm returns a warning callback when a spoof is detected
ENUM_IBSU_PROPERTY_VIEW_ENCRYPTION_↩ IMAGE_MODE	View encrypt Image The default is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_FINGERPRINT_↩ SEGMENTATION_MODE	Select fingerprint segmentation mode The default is 0. [Get and set.]
ENUM_IBSU_PROPERTY_ROLL_METHOD	Enhanced roll Method The default value is 0. [Get and set.]
ENUM_IBSU_PROPERTY_RENEWAL_↩ OPPOSITE_IMGAE_LEVEL	Select a level of opposite image value during roll The default value is 0. [Get and set.] 0 : No use 1 : renewal if roll image is moved as 1.2mm. 2 : renewal if roll image is moved as 2.4mm. 3 : renewal if roll image is moved as 3.6mm.
ENUM_IBSU_PROPERTY_PREVIEW_IMAGE_↩ QUALITY_FOR_KOJAK	Enable High quality preview image for Kojak The default value is 0. [Get and set.]
ENUM_IBSU_PROPERTY_ADAPTIVE_↩ CAPTURE_MODE	Enable Adaptive Capture The default value is TRUE. [Get only.]
ENUM_IBSU_PROPERTY_ENABLE_KOJAK_↩ BEHAVIOR_2_6	Enable Kojak 2.6 behavior The default value is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_DISABLE_SEGMENT↩ _ROTATION	Disable Segment rectangles rotation The default value is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_DR_MODE_ZOOM_IN	Enable Zoom-In mode : It sets Zoom-In when fingers start being detected in DR mode. The default value is TRUE. This property only set FALSE in WINDOWS. Can't set FALSE in Linux or Android or Windows JNI-JAVA environments. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_1	Reserved for manufacturer strings. [Need a reserve code]
ENUM_IBSU_PROPERTY_RESERVED_2	
ENUM_IBSU_PROPERTY_RESERVED_100	

Enumerator

ENUM_IBSU_PROPERTY_RESERVED_IMAGE_↩ PROCESS_THRESHOLD	The preview image 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_ENABLE↩ _TOF_FOR_ROLL	Enable TOF for roll capture The default is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_↩ CAPTURE_BRIGHTNESS_THRESHOLD_FOR_↩ FLAT	Change brightness threshold for flat capture The default values are dependant on the scanner. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_↩ CAPTURE_BRIGHTNESS_THRESHOLD_FOR_↩ ROLL	Change brightness threshold for roll capture The default values are dependant on the scanner. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_↩ ENHANCED_RESULT_IMAGE	Change result image to be enhanced The default value is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_↩ ENHANCED_RESULT_IMAGE_LEVEL	Select a level of image processing The default value is 0. [Get and set.] 0 : Minimum enhancement 5 : Maximum enhancement
ENUM_IBSU_PROPERTY_RESERVED_ENABLE↩ _SLIP_DETECTION	Enable Slip detection The default value is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_SLIP_↩ DETECTION_LEVEL	Change level of Slip detection The default value is 3. The Range is from 1 to 10. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_ENABLE↩ _TRICK_CAPTURE	Enable Trick Capture The default value is TRUE. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_ENABLE↩ _CBP_MODE	Enable to set CBP operation mode The default value is FALSE. [Get and set.]
ENUM_IBSU_PROPERTY_RESERVED_RAW_↩ IMAGE_WIDTH	
ENUM_IBSU_PROPERTY_RESERVED_RAW_↩ IMAGE_HEIGHT	
ENUM_IBSU_PROPERTY_RESERVED_TFT_↩ NOISE_REMOVAL	
ENUM_IBSU_PROPERTY_RESERVED_LINE_↩ BLACK_FILL	
ENUM_IBSU_PROPERTY_RESERVED_↩ RECALCILATE_BRIGHTNESS	
ENUM_IBSU_PROPERTY_RESERVED_LINE_↩ RESTORE	
ENUM_IBSU_PROPERTY_RESERVED_SW_↩ UNIFORMITY	
ENUM_IBSU_PROPERTY_RESERVED_SET_↩ ROLL_TEST_MODE	

Definition at line 654 of file IBScanUltimateApi_defs.h.

```
00655 {
00657     ENUM_IBSU_PROPERTY_PRODUCT_ID,
00658
00660     ENUM_IBSU_PROPERTY_SERIAL_NUMBER,
00661
00663     ENUM_IBSU_PROPERTY_VENDOR_ID,
00664
00666     ENUM_IBSU_PROPERTY_IBIA_VENDOR_ID,
```

```
00667
00669     ENUM_IBSU_PROPERTY_IBIA_VERSION,
00670
00672     ENUM_IBSU_PROPERTY_IBIA_DEVICE_ID,
00673
00675     ENUM_IBSU_PROPERTY_FIRMWARE,
00676
00678     ENUM_IBSU_PROPERTY_REVISION,
00679
00681     ENUM_IBSU_PROPERTY_PRODUCTION_DATE,
00682
00684     ENUM_IBSU_PROPERTY_SERVICE_DATE,
00685
00687     ENUM_IBSU_PROPERTY_IMAGE_WIDTH,
00688
00690     ENUM_IBSU_PROPERTY_IMAGE_HEIGHT,
00691
00695     ENUM_IBSU_PROPERTY_IGNORE_FINGER_TIME,
00696
00698     ENUM_IBSU_PROPERTY_RECOMMENDED_LEVEL,
00699
00701     ENUM_IBSU_PROPERTY_POLLINGTIME_TO_BGETIMAGE,
00702
00704     ENUM_IBSU_PROPERTY_ENABLE_POWER_SAVE_MODE,
00705
00708     ENUM_IBSU_PROPERTY_RETRY_WRONG_COMMUNICATION,
00709
00713     ENUM_IBSU_PROPERTY_CAPTURE_TIMEOUT,
00714
00717     ENUM_IBSU_PROPERTY_ROLL_MIN_WIDTH,
00718
00722     ENUM_IBSU_PROPERTY_ROLL_MODE,
00723
00728     ENUM_IBSU_PROPERTY_ROLL_LEVEL,
00729
00733     ENUM_IBSU_PROPERTY_CAPTURE_AREA_THRESHOLD,
00734
00737     ENUM_IBSU_PROPERTY_ENABLE_DECIMATION,
00738
00742     ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_ON_RELEASE,
00743
00745     ENUM_IBSU_PROPERTY_DEVICE_INDEX,
00746
00748     ENUM_IBSU_PROPERTY_DEVICE_ID,
00749
00755     ENUM_IBSU_PROPERTY_SUPER_DRY_MODE,
00756
00761     ENUM_IBSU_PROPERTY_MIN_CAPTURE_TIME_IN_SUPER_DRY_MODE,
00762
00764     ENUM_IBSU_PROPERTY_ROLLED_IMAGE_WIDTH,
00765
00767     ENUM_IBSU_PROPERTY_ROLLED_IMAGE_HEIGHT,
00768
00771     ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE,
00772
00775     ENUM_IBSU_PROPERTY_ROLL_IMAGE_OVERRIDE,
00776
00779     ENUM_IBSU_PROPERTY_WARNING_MESSAGE_INVALID_AREA,
00780
00783     ENUM_IBSU_PROPERTY_ENABLE_WET_FINGER_DETECT,
00784
00789     ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL,
00790
00795     ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL_THRESHOLD,
00796
00801     ENUM_IBSU_PROPERTY_START_POSITION_OF_ROLLING_AREA,
00802
00805     ENUM_IBSU_PROPERTY_START_ROLL_WITHOUT_LOCK,
00806
00809     ENUM_IBSU_PROPERTY_ENABLE_TOF,
00810
00813     ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION,
00814
00816     ENUM_IBSU_PROPERTY_IS_SPOOF_SUPPORTED,
00817
00820         ENUM_IBSU_PROPERTY_ENABLE_SPOOF,
00821
00836     ENUM_IBSU_PROPERTY_SPOOF_LEVEL,
00837
00840     ENUM_IBSU_PROPERTY_VIEW_ENCRYPTION_IMAGE_MODE,
00841
00844     ENUM_IBSU_PROPERTY_FINGERPRINT_SEGMENTATION_MODE,
00845
00848     ENUM_IBSU_PROPERTY_ROLL_METHOD,
00849
00857     ENUM_IBSU_PROPERTY_RENEWAL_OPPOSITE_IMGAE_LEVEL,
00858
```

```
00861     ENUM_IBSU_PROPERTY_PREVIEW_IMAGE_QUALITY_FOR_KOJAK,
00862
00865     ENUM_IBSU_PROPERTY_ADAPTIVE_CAPTURE_MODE,
00866
00869     ENUM_IBSU_PROPERTY_ENABLE_KOJAK_BEHAVIOR_2_6,
00870
00873     ENUM_IBSU_PROPERTY_DISABLE_SEGMENT_ROTATION,
00874
00878     ENUM_IBSU_PROPERTY_DR_MODE_ZOOM_IN,
00879
00881     ENUM_IBSU_PROPERTY_RESERVED_1 = 200,
00882     ENUM_IBSU_PROPERTY_RESERVED_2,
00883     ENUM_IBSU_PROPERTY_RESERVED_100,
00884
00892     ENUM_IBSU_PROPERTY_RESERVED_IMAGE_PROCESS_THRESHOLD = 400,
00893
00896     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TOF_FOR_ROLL,
00897
00900     ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_FLAT,
00901
00904     ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_ROLL,
00905
00908     ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE,
00909
00914     ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE_LEVEL,
00915
00918     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_SLIP_DETECTION,
00919
00922     ENUM_IBSU_PROPERTY_RESERVED_SLIP_DETECTION_LEVEL,
00923
00926     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TRICK_CAPTURE,
00927
00930     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_CBP_MODE,
00931
00932     /* Raw Image width value.  [Get only.] */
00933     ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_WIDTH,
00934
00935     /* Raw Image height value.  [Get only.] */
00936     ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_HEIGHT,
00937
00938
00939     ENUM_IBSU_PROPERTY_RESERVED_TFT_NOISE_REMOVAL,
00940
00941     ENUM_IBSU_PROPERTY_RESERVED_LINE_BLACK_FILL,
00942
00943     ENUM_IBSU_PROPERTY_RESERVED_RECALCILATE_BRIGHTNESS,
00944
00945     ENUM_IBSU_PROPERTY_RESERVED_LINE_RESTORE,
00946
00947     ENUM_IBSU_PROPERTY_RESERVED_SW_UNIFORMITY,
00948
00949     ENUM_IBSU_PROPERTY_RESERVED_SET_ROLL_TEST_MODE,
00950 }
```

12.28 Enumeration - Property ID for Client Window

Enumeration of client window property IDs. Properties can be retrieved with IBSU_GetClientWindowProperty(); some properties can also be set with IBSU_SetClientWindowProperty().

Enumerations

- enum IBSU_ClientWindowPropertyId {
  ENUM_IBSU_WINDOW_PROPERTY_BK_COLOR , ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE
  , ENUM_IBSU_WINDOW_PROPERTY_DISP_INVALID_AREA , ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR
  ,
  ENUM_IBSU_WINDOW_PROPERTY_LEFT_MARGIN , ENUM_IBSU_WINDOW_PROPERTY_TOP_MARGIN
  , ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_WIDTH , ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTO
  ,
  ENUM_IBSU_WINDOW_PROPERTY_KEEP_REDRAW_LAST_IMAGE , ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE
  }

12.28.1 Detailed Description

Enumeration of client window property IDs. Properties can be retrieved with IBSU_GetClientWindowProperty(); some properties can also be set with IBSU_SetClientWindowProperty().

12.28.2 Enumeration Type Documentation

12.28.2.1 IBSU_ClientWindowPropertyId

```
enum IBSU_ClientWindowPropertyId
```

Enumerator

ENUM_IBSU_WINDOW_PROPERTY_BK_COLOR	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_ROLL_↩ GUIDE_LINE	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_DISP_↩ INVALID_AREA	Draw arrow to display invalid area (TRUE or FALSE). The default is FALSE. [Get and set.]
ENUM_IBSU_WINDOW_PROPERTY_SCALE_↩ FACTOR	Get the scale of the display image on client window, as a floating point value.
ENUM_IBSU_WINDOW_PROPERTY_LEFT_↩ MARGIN	Get the left margin of the displayed image in relation to the client window, as an integer.
ENUM_IBSU_WINDOW_PROPERTY_TOP_MARGIN	Get the top margin of the displayed image in relation to the client window, as an integer.
ENUM_IBSU_WINDOW_PROPERTY_ROLL_↩ GUIDE_LINE_WIDTH	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_SCALE_↩ FACTOR_EX	Get the extended scale of the display image on client window, as a integer value.
ENUM_IBSU_WINDOW_PROPERTY_KEEP_↩ REDRAW_LAST_IMAGE	Keep the last image for redrawing of the display image when calling IBSU_RedrawClientWindow. (TRUE to enable or FALSE to disable). The default is TRUE. [Get and set.]
ENUM_IBSU_WINDOW_PROPERTY_ROLL_↩ GUIDE_LINE_COLOR	Color of the roll guide line. The valid range is between 0x00000000 and 0xFFFFFFFF, inclusive, with the default of 0x0000FF (Red Color). [Get and set.]

Definition at line 968 of file IBScanUltimateApi_defs.h.

```
00969 {
00972     ENUM_IBSU_WINDOW_PROPERTY_BK_COLOR,
00973
```

```
00976     ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE,
00977
00979     ENUM_IBSU_WINDOW_PROPERTY_DISP_INVALID_AREA,
00980
00982     ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR,
00983
00985     ENUM_IBSU_WINDOW_PROPERTY_LEFT_MARGIN,
00986
00988     ENUM_IBSU_WINDOW_PROPERTY_TOP_MARGIN,
00989
00992     ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_WIDTH,
00993
00995     ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR_EX,
00996
00999     ENUM_IBSU_WINDOW_PROPERTY_KEEP_REDRAW_LAST_IMAGE,
01000
01003     ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_COLOR,
01004 }
```

12.29 Enumeration - Finger Count State

Enumeration of finger count states.

Enumerations

- enum IBSU_FingerCountState { ENUM_IBSU_FINGER_COUNT_OK , ENUM_IBSU_TOO_MANY_FINGERS , ENUM_IBSU_TOO_FEW_FINGERS , ENUM_IBSU_NON_FINGER }

12.29.1 Detailed Description

Enumeration of finger count states.

12.29.2 Enumeration Type Documentation

12.29.2.1 IBSU_FingerCountState

```
enum IBSU_FingerCountState
```

Enumerator

ENUM_IBSU_FINGER_COUNT_OK	
ENUM_IBSU_TOO_MANY_FINGERS	
ENUM_IBSU_TOO_FEW_FINGERS	
ENUM_IBSU_NON_FINGER	

Definition at line 1021 of file IBScanUltimateApi_defs.h.

```
01022 {
01023     ENUM_IBSU_FINGER_COUNT_OK,
01024     ENUM_IBSU_TOO_MANY_FINGERS,
01025     ENUM_IBSU_TOO_FEW_FINGERS,
01026     ENUM_IBSU_NON_FINGER
01027 }
```

12.30 Enumeration - Finger Count State

Enumeration of finger quality states.

Enumerations

- enum IBSU_FingerQualityState {
  ENUM_IBSU_FINGER_NOT_PRESENT , ENUM_IBSU_QUALITY_GOOD , ENUM_IBSU_QUALITY_FAIR , ENUM_IBSU_QUALITY_POOR ,
  ENUM_IBSU_QUALITY_INVALID_AREA_TOP , ENUM_IBSU_QUALITY_INVALID_AREA_LEFT ,
  ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT , ENUM_IBSU_QUALITY_INVALID_AREA_BOTTOM }

12.30.1 Detailed Description

Enumeration of finger quality states.

12.30.2 Enumeration Type Documentation

12.30.2.1 IBSU_FingerQualityState

```
enum IBSU_FingerQualityState
```

Enumerator

ENUM_IBSU_FINGER_NOT_PRESENT	
ENUM_IBSU_QUALITY_GOOD	
ENUM_IBSU_QUALITY_FAIR	
ENUM_IBSU_QUALITY_POOR	
ENUM_IBSU_QUALITY_INVALID_AREA_TOP	Finger position is not valid on top side.
ENUM_IBSU_QUALITY_INVALID_AREA_LEFT	Finger position is not valid on left side.
ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT	Finger position is not valid on right side.
ENUM_IBSU_QUALITY_INVALID_AREA_BOTTOM	Finger position is not valid on bottom side.

Definition at line 1044 of file IBScanUltimateApi_defs.h.

```
01045 {
01046     ENUM_IBSU_FINGER_NOT_PRESENT,
01047     ENUM_IBSU_QUALITY_GOOD,
01048     ENUM_IBSU_QUALITY_FAIR,
01049     ENUM_IBSU_QUALITY_POOR,
01051     ENUM_IBSU_QUALITY_INVALID_AREA_TOP,
01053     ENUM_IBSU_QUALITY_INVALID_AREA_LEFT,
01055     ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT,
01057     ENUM_IBSU_QUALITY_INVALID_AREA_BOTTOM
01058 }
```

12.31 Enumeration - LE Operation Mode

Enumeration of light emitting (LE) file operation modes.

Enumerations

- enum IBSU_LEOperationMode { ENUM_IBSU_LE_OPERATION_AUTO , ENUM_IBSU_LE_OPERATION_ON , ENUM_IBSU_LE_OPERATION_OFF }

12.31.1 Detailed Description

Enumeration of light emitting (LE) file operation modes.

12.31.2 Enumeration Type Documentation

12.31.2.1 IBSU_LEOperationMode

```
enum IBSU_LEOperationMode
```

Enumerator

ENUM_IBSU_LE_OPERATION_AUTO	
ENUM_IBSU_LE_OPERATION_ON	
ENUM_IBSU_LE_OPERATION_OFF	

Definition at line 1075 of file IBScanUltimateApi_defs.h.

```
01076 {
01077     ENUM_IBSU_LE_OPERATION_AUTO,
01078     ENUM_IBSU_LE_OPERATION_ON,
01079     ENUM_IBSU_LE_OPERATION_OFF
01080 }
```

12.32 Enumeration - Platen State

Enumeration of platen states.

Enumerations

- enum IBSU_PlatenState { ENUM_IBSU_PLATEN_CLEARD , ENUM_IBSU_PLATEN_HAS_FINGERS }

12.32.1 Detailed Description

Enumeration of platen states.

12.32.2 Enumeration Type Documentation

12.32.2.1 IBSU_PlatenState

```
enum IBSU_PlatenState
```

Enumerator

ENUM_IBSU_PLATEN_CLEARD	
ENUM_IBSU_PLATEN_HAS_FINGERS	

Definition at line 1097 of file IBScanUltimateApi_defs.h.

```
01098 {
01099     ENUM_IBSU_PLATEN_CLEARD,
01100     ENUM_IBSU_PLATEN_HAS_FINGERS
01101 }
```

12.33 Enumeration - Callback Events

Enumeration of events that can trigger callbacks.

Enumerations

- enum IBSU_Events {
  ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT , ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK , ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE , ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION , ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION , ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE , ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY , ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT , ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS , ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE , ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE , ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE , ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE_EX , ENUM_IBSU_ESSENTIAL_EVENT_KEYBUTTON
  }

12.33.1 Detailed Description

Enumeration of events that can trigger callbacks.

12.33.2 Enumeration Type Documentation

12.33.2.1 IBSU_Events

```
enum IBSU_Events
```

Enumerator

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

Enumerator

ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_↩ IMAGE	Callback when a new preview image is available from a device.
ENUM_IBSU_ESSENTIAL_EVENT_TAKING_↩ ACQUISITION	Callback for rolled print acquisition when rolling should begin.
ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_↩ ACQUISITION	Callback for rolled print acquisition when rolling completes.
ENUM_IBSU_ESSENTIAL_EVENT_RESULT_↩ IMAGE	Callback when result image is available for a capture (deprecated).
ENUM_IBSU_OPTIONAL_EVENT_FINGER_↩ QUALITY	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 the platen was not clear when capture started or has since become clear.
ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_↩ OPEN_DEVICE	Callback when asynchronous device initialization completes.
ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_↩ MESSAGE	Callback when a warning message is generated.
ENUM_IBSU_ESSENTIAL_EVENT_RESULT_↩ IMAGE_EX	Callback when the result image is available for a capture (with extended information).
ENUM_IBSU_ESSENTIAL_EVENT_KEYBUTTON	Callback when key buttons are pressed

Definition at line 1118 of file IBScanUltimateApi_defs.h.

```
01119 {
01121     ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT,
01122
01124     ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK,
01125
01127     ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE,
01128
01130     ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION,
01131
01133     ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION,
01134
01136     ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE,
01137
01139     ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY,
01140
01142     ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT,
01143
01145     ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS,
01146
01148     ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE,
01149
01151     ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE,
01152
01154     ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE,
01155
01157     ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE_EX,
01158
01160     ENUM_IBSU_ESSENTIAL_EVENT_KEYBUTTON
01161 }
```

12.34 Enumeration - LED Type

Enumeration of LED types.

Enumerations

- enum IBSU_LedType { ENUM_IBSU_LED_TYPE_NONE , ENUM_IBSU_LED_TYPE_TSCAN , ENUM_IBSU_LED_TYPE_FSC
  }

12.34.1 Detailed Description

Enumeration of LED types.

12.34.2 Enumeration Type Documentation

12.34.2.1 IBSU_LedType

```
enum IBSU_LedType
```

Enumerator

ENUM_IBSU_LED_TYPE_NONE	No LED field.
ENUM_IBSU_LED_TYPE_TSCAN	Two-scanner type (e.g., Watson).
ENUM_IBSU_LED_TYPE_FSCAN	Four-scanner type (e.g., Kojak).

Definition at line 1178 of file IBScanUltimateApi_defs.h.

```
01179 {
01181     ENUM_IBSU_LED_TYPE_NONE,
01182
01184     ENUM_IBSU_LED_TYPE_TSCAN,
01185
01187     ENUM_IBSU_LED_TYPE_FSCAN
01188 }
```

12.35 Enumeration - Rolling State

Enumeration of rolling print acquisition states.

Enumerations

- enum IBSU_RollingState { ENUM_IBSU_ROLLING_NOT_PRESENT , ENUM_IBSU_ROLLING_TAKE_ACQUISITION
  , ENUM_IBSU_ROLLING_COMPLETE_ACQUISITION , ENUM_IBSU_ROLLING_RESULT_IMAGE }

12.35.1 Detailed Description

Enumeration of rolling print acquisition states.

12.35.2 Enumeration Type Documentation

12.35.2.1 IBSU_RollingState

```
enum IBSU_RollingState
```

Enumerator

ENUM_IBSU_ROLLING_NOT_PRESENT	
ENUM_IBSU_ROLLING_TAKE_ACQUISITION	
ENUM_IBSU_ROLLING_COMPLETE_ACQUISITION	
ENUM_IBSU_ROLLING_RESULT_IMAGE	

Definition at line 1205 of file IBScanUltimateApi_defs.h.

```
01206 {
01207     ENUM_IBSU_ROLLING_NOT_PRESENT,
01208     ENUM_IBSU_ROLLING_TAKE_ACQUISITION,
01209     ENUM_IBSU_ROLLING_COMPLETE_ACQUISITION,
01210     ENUM_IBSU_ROLLING_RESULT_IMAGE
01211 }
```

12.36 Enumeration - Client Window - Overlay Pattern Type

Enumeration of the shape pattern to use for the overlay on client window.

Enumerations

- enum IBSU_OverlayShapePattern { ENUM_IBSU_OVERLAY_SHAPE_RECTANGLE , ENUM_IBSU_OVERLAY_SHAPE_ELLI , ENUM_IBSU_OVERLAY_SHAPE_CROSS , ENUM_IBSU_OVERLAY_SHAPE_ARROW }

12.36.1 Detailed Description

Enumeration of the shape pattern to use for the overlay on client window.

12.36.2 Enumeration Type Documentation

12.36.2.1 IBSU_OverlayShapePattern

```
enum IBSU_OverlayShapePattern
```

Enumerator

ENUM_IBSU_OVERLAY_SHAPE_RECTANGLE	
ENUM_IBSU_OVERLAY_SHAPE_ELLIPSE	
ENUM_IBSU_OVERLAY_SHAPE_CROSS	
ENUM_IBSU_OVERLAY_SHAPE_ARROW	

Definition at line 1228 of file IBScanUltimateApi_defs.h.

```
01229 {
01230     ENUM_IBSU_OVERLAY_SHAPE_RECTANGLE,
01231     ENUM_IBSU_OVERLAY_SHAPE_ELLIPSE,
01232     ENUM_IBSU_OVERLAY_SHAPE_CROSS,
01233     ENUM_IBSU_OVERLAY_SHAPE_ARROW
01234 }
```

12.37 Enumeration - Combine Two Finger Image Type

Enumeration of hand to use for combining two images into one.

Enumerations

- enum IBSU_CombineImageWhichHand { ENUM_IBSU_COMBINE_IMAGE_LEFT_HAND , ENUM_IBSU_COMBINE_IMAGE_
  }

12.37.1 Detailed Description

Enumeration of hand to use for combining two images into one.

12.37.2 Enumeration Type Documentation

12.37.2.1 IBSU_CombineImageWhichHand

```
enum IBSU_CombineImageWhichHand
```

Enumerator

ENUM_IBSU_COMBINE_IMAGE_LEFT_HAND	
ENUM_IBSU_COMBINE_IMAGE_RIGHT_HAND	

Definition at line 1251 of file IBScanUltimateApi_defs.h.

```
01252 {
01253     ENUM_IBSU_COMBINE_IMAGE_LEFT_HAND,
01254     ENUM_IBSU_COMBINE_IMAGE_RIGHT_HAND
01255 }
```

12.38 Enumeration - Beeper Type

Enumeration of Beeper types.

Typedefs

- typedef enum enumIBSU_BeeperType IBSU_BeeperType

Enumerations

- enum enumIBSU_BeeperType { ENUM_IBSU_BEEPER_TYPE_NONE , ENUM_IBSU_BEEPER_TYPE_MONOTONE }

12.38.1 Detailed Description

Enumeration of Beeper types.

12.38.2 Typedef Documentation

12.38.2.1 IBSU_BeeperType

```
typedef enum enumIBSU_BeeperType IBSU_BeeperType
```

12.38.3 Enumeration Type Documentation

12.38.3.1 enumIBSU_BeeperType

```
enum enumIBSU_BeeperType
```

Enumerator

ENUM_IBSU_BEEPER_TYPE_NONE	No Beeper field.
ENUM_IBSU_BEEPER_TYPE_MONOTONE	Monotone type.

Definition at line 1272 of file IBScanUltimateApi_defs.h.

```
01273 {
01275     ENUM_IBSU_BEEPER_TYPE_NONE,
01276
01278     ENUM_IBSU_BEEPER_TYPE_MONOTONE,
01279 }
```

12.39 Enumeration - Beeper Pattern

Enumeration of the beep pattern.

Enumerations

- enum IBSU_BeepPattern { ENUM_IBSU_BEEP_PATTERN_GENERIC , ENUM_IBSU_BEEP_PATTERN_REPEAT }

12.39.1 Detailed Description

Enumeration of the beep pattern.

12.39.2 Enumeration Type Documentation

12.39.2.1 IBSU_BeepPattern

```
enum IBSU_BeepPattern
```

Enumerator

ENUM_IBSU_BEEP_PATTERN_GENERIC	Generic type.
ENUM_IBSU_BEEP_PATTERN_REPEAT	Repeat type.

Definition at line 1296 of file IBScanUltimateApi_defs.h.

```
01297 {
01299     ENUM_IBSU_BEEP_PATTERN_GENERIC,
01300
01302     ENUM_IBSU_BEEP_PATTERN_REPEAT
01303 }
```

12.40 Enumeration - Encryption Mode

Enumeration of the Encryption Mode.

Enumerations

- enum IBSU_EncryptionMode { ENUM_IBSU_ENCRYPTION_KEY_RANDOM , ENUM_IBSU_ENCRYPTION_KEY_CUSTOM , ENUM_IBSU_ENCRYPTION_KEY_DEFAULT }

12.40.1 Detailed Description

Enumeration of the Encryption Mode.

12.40.2 Enumeration Type Documentation

12.40.2.1 IBSU_EncryptionMode

```
enum IBSU_EncryptionMode
```

Enumerator

ENUM_IBSU_ENCRYPTION_KEY_RANDOM	Random Key generated by own library.
ENUM_IBSU_ENCRYPTION_KEY_CUSTOM	Custom Key provided by user.
ENUM_IBSU_ENCRYPTION_KEY_DEFAULT	Default Key for Development (only for IB Engineer)

Definition at line 1320 of file IBScanUltimateApi_defs.h.

```
01321 {
01323     ENUM_IBSU_ENCRYPTION_KEY_RANDOM,
01324
01326     ENUM_IBSU_ENCRYPTION_KEY_CUSTOM,
01327
01329     ENUM_IBSU_ENCRYPTION_KEY_DEFAULT
01330 }
```

12.41 Enumeration - Matcher - Image format

Enumeration of image formats to support IBScanMatcher integration.

Enumerations

- enum IBSM_ImageFormat { IBSM_IMG_FORMAT_NO_BIT_PACKING =0 , IBSM_IMG_FORMAT_BIT_PACKED , IBSM_IMG_FORMAT_WSQ , IBSM_IMG_FORMAT_JPEG_LOSSY , IBSM_IMG_FORMAT_JPEG2000_LOSSY , IBSM_IMG_FORMAT_JPEG2000_LOSSLESS , IBSM_IMG_FORMAT_PNG , IBSM_IMG_FORMAT_UNKNOWN }

12.41.1 Detailed Description

Enumeration of image formats to support IBScanMatcher integration.

12.41.2 Enumeration Type Documentation

12.41.2.1 IBSM_ImageFormat

```
enum IBSM_ImageFormat
```

Enumerator

IBSM_IMG_FORMAT_NO_BIT_PACKING	
IBSM_IMG_FORMAT_BIT_PACKED	
IBSM_IMG_FORMAT_WSQ	
IBSM_IMG_FORMAT_JPEG_LOSSY	
IBSM_IMG_FORMAT_JPEG2000_LOSSY	
IBSM_IMG_FORMAT_JPEG2000_LOSSLESS	
IBSM_IMG_FORMAT_PNG	
IBSM_IMG_FORMAT_UNKNOWN	

Definition at line 1347 of file IBScanUltimateApi_defs.h.

```
01348 {
01349     IBSM_IMG_FORMAT_NO_BIT_PACKING=0,
01350     IBSM_IMG_FORMAT_BIT_PACKED,
01351     IBSM_IMG_FORMAT_WSQ,
01352     IBSM_IMG_FORMAT_JPEG_LOSSY,
01353     IBSM_IMG_FORMAT_JPEG2000_LOSSY,
01354     IBSM_IMG_FORMAT_JPEG2000_LOSSLESS,
01355     IBSM_IMG_FORMAT_PNG,
01356     IBSM_IMG_FORMAT_UNKNOWN
01357 }
```

12.42 Enumeration - Matcher - Image Impresstion Type

Enumeration of image impression types to support IBScanMatcher integration.

Enumerations

- enum IBSM_ImpressionType {
  IBSM_IMPRESSION_TYPE_LIVE_SCAN_PLAIN =0 , IBSM_IMPRESSION_TYPE_LIVE_SCAN_ROLLED ,
  IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PLAIN , IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_ROLLED
  ,
  IBSM_IMPRESSION_TYPE_LATENT_IMPRESSION , IBSM_IMPRESSION_TYPE_LATENT_TRACING ,
  IBSM_IMPRESSION_TYPE_LATENT_PHOTO , IBSM_IMPRESSION_TYPE_LATENT_LIFT ,
  IBSM_IMPRESSION_TYPE_LIVE_SCAN_SWIPE , IBSM_IMPRESSION_TYPE_LIVE_SCAN_VERTICAL_ROLL
  , IBSM_IMPRESSION_TYPE_LIVE_SCAN_PALM , IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PALM ,
  IBSM_IMPRESSION_TYPE_LATENT_PALM_IMPRESSION , IBSM_IMPRESSION_TYPE_LATENT_PALM_TRACING
  , IBSM_IMPRESSION_TYPE_LATENT_PALM_PHOTO , IBSM_IMPRESSION_TYPE_LATENT_PALM_LIFT
  ,
  IBSM_IMPRESSION_TYPE_LIVE_SCAN_OPTICAL_CONTRCTLESS_PLAIN =24 , IBSM_IMPRESSION_TYPE_OTHER
  =28 , IBSM_IMPRESSION_TYPE_UNKNOWN =29 }

12.42.1 Detailed Description

Enumeration of image impression types to support IBScanMatcher integration.

12.42.2 Enumeration Type Documentation

12.42.2.1 IBSM_ImpressionType

```
enum IBSM_ImpressionType
```

Enumerator

IBSM_IMPRESSION_TYPE_LIVE_SCAN_PLAIN	
IBSM_IMPRESSION_TYPE_LIVE_SCAN_ROLLED	
IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PLAIN	
IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_ROLLED	
IBSM_IMPRESSION_TYPE_LATENT_IMPRESSION	
IBSM_IMPRESSION_TYPE_LATENT_TRACING	
IBSM_IMPRESSION_TYPE_LATENT_PHOTO	
IBSM_IMPRESSION_TYPE_LATENT_LIFT	
IBSM_IMPRESSION_TYPE_LIVE_SCAN_SWIPE	
IBSM_IMPRESSION_TYPE_LIVE_SCAN_VERTICAL_ROLL	
IBSM_IMPRESSION_TYPE_LIVE_SCAN_PALM	
IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PALM	
IBSM_IMPRESSION_TYPE_LATENT_PALM_IMPRESSION	
IBSM_IMPRESSION_TYPE_LATENT_PALM_TRACING	
IBSM_IMPRESSION_TYPE_LATENT_PALM_PHOTO	
IBSM_IMPRESSION_TYPE_LATENT_PALM_LIFT	
IBSM_IMPRESSION_TYPE_LIVE_SCAN_OPTICAL_CONTRCTLESS_PLAIN	
IBSM_IMPRESSION_TYPE_OTHER	
IBSM_IMPRESSION_TYPE_UNKNOWN	

Definition at line 1374 of file IBScanUltimateApi_defs.h.

```
01375 {
01376     IBSM_IMPRESSION_TYPE_LIVE_SCAN_PLAIN=0,
01377     IBSM_IMPRESSION_TYPE_LIVE_SCAN_ROLLED,
01378     IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PLAIN,
01379     IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_ROLLED,
01380     IBSM_IMPRESSION_TYPE_LATENT_IMPRESSION,
01381     IBSM_IMPRESSION_TYPE_LATENT_TRACING,
01382     IBSM_IMPRESSION_TYPE_LATENT_PHOTO,
01383     IBSM_IMPRESSION_TYPE_LATENT_LIFT,
01384     IBSM_IMPRESSION_TYPE_LIVE_SCAN_SWIPE,
01385     IBSM_IMPRESSION_TYPE_LIVE_SCAN_VERTICAL_ROLL,
01386     IBSM_IMPRESSION_TYPE_LIVE_SCAN_PALM,
01387     IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PALM,
01388     IBSM_IMPRESSION_TYPE_LATENT_PALM_IMPRESSION,
01389     IBSM_IMPRESSION_TYPE_LATENT_PALM_TRACING,
01390     IBSM_IMPRESSION_TYPE_LATENT_PALM_PHOTO,
01391     IBSM_IMPRESSION_TYPE_LATENT_PALM_LIFT,
01392     IBSM_IMPRESSION_TYPE_LIVE_SCAN_OPTICAL_CONTRCTLESS_PLAIN=24,
01393     IBSM_IMPRESSION_TYPE_OTHER=28,
01394     IBSM_IMPRESSION_TYPE_UNKNOWN=29
01395 }
```

12.43 Enumeration - Matcher - Finger Position

Enumeration of finger positions to support IBScanMatcher integration.

Typedefs

- typedef enum enum_IBSM_FingerPosition IBSM_FingerPosition

Enumerations

- enum enum_IBSM_FingerPosition {
  IBSM_FINGER_POSITION_UNKNOWN =0 , IBSM_FINGER_POSITION_RIGHT_THUMB , IBSM_FINGER_POSITION_RIGH
  , 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
  ,
  IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE_AND_RING , IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_R
  , IBSM_FINGER_POSITION_UNKNOWN_HALF_PALM , IBSM_FINGER_POSITION_RIGHT_HALF_PALM
  ,
  IBSM_FINGER_POSITION_LEFT_HALF_PALM , IBSM_FINGER_POSITION_RIGHT_LOWER_HALF_PALM
  , IBSM_FINGER_POSITION_RIGHT_UPPER_HALF_PALM , IBSM_FINGER_POSITION_LEFT_LOWER_HALF_PALM
  ,
  IBSM_FINGER_POSITION_LEFT_UPPER_HALF_PALM , IBSM_FINGER_POSITION_LEFT_CENTER_JOINT
  , IBSM_FINGER_POSITION_RIGHT_CENTER_JOINT , IBSM_FINGER_POSITION_LEFT_SIDE_JOINT ,
  IBSM_FINGER_POSITION_RIGHT_SIDE_JOINT , IBSM_FINGER_POSITION_LEFT_ROLL_JOINT ,
  IBSM_FINGER_POSITION_RIGHT_ROLL_JOINT , IBSM_FINGER_POSITION_LEFT_ROLL_UP ,
  IBSM_FINGER_POSITION_RIGHT_ROLL_UP }

12.43.1 Detailed Description

Enumeration of finger positions to support IBScanMatcher integration.

12.43.2 Typedef Documentation

12.43.2.1 IBSM_FingerPosition

```
typedef enum enum_IBSM_FingerPosition IBSM_FingerPosition
```

12.43.3 Enumeration Type Documentation

12.43.3.1 enum_IBSM_FingerPosition

```
enum enum_IBSM_FingerPosition
```

Enumerator

IBSM_FINGER_POSITION_UNKNOWN	
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	
IBSM_FINGER_POSITION_PLAIN_LEFT_FOUR_FINGERS	
IBSM_FINGER_POSITION_PLAIN_THUMBS	
IBSM_FINGER_POSITION_UNKNOWN_PALM	
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	
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	

Enumerator

IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE_AND_RING	
IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_RING_AND_LITTLE	
IBSM_FINGER_POSITION_UNKNOWN_HALF_PALM	
IBSM_FINGER_POSITION_RIGHT_HALF_PALM	
IBSM_FINGER_POSITION_LEFT_HALF_PALM	
IBSM_FINGER_POSITION_RIGHT_LOWER_HALF_PALM	
IBSM_FINGER_POSITION_RIGHT_UPPER_HALF_PALM	
IBSM_FINGER_POSITION_LEFT_LOWER_HALF_PALM	
IBSM_FINGER_POSITION_LEFT_UPPER_HALF_PALM	
IBSM_FINGER_POSITION_LEFT_CENTER_JOINT	
IBSM_FINGER_POSITION_RIGHT_CENTER_JOINT	
IBSM_FINGER_POSITION_LEFT_SIDE_JOINT	
IBSM_FINGER_POSITION_RIGHT_SIDE_JOINT	
IBSM_FINGER_POSITION_LEFT_ROLL_JOINT	
IBSM_FINGER_POSITION_RIGHT_ROLL_JOINT	
IBSM_FINGER_POSITION_LEFT_ROLL_UP	
IBSM_FINGER_POSITION_RIGHT_ROLL_UP	

Definition at line 1412 of file IBScanUltimateApi_defs.h.

```
01413 {
01414     IBSM_FINGER_POSITION_UNKNOWN=0,
01415     IBSM_FINGER_POSITION_RIGHT_THUMB,
01416     IBSM_FINGER_POSITION_RIGHT_INDEX_FINGER,
01417     IBSM_FINGER_POSITION_RIGHT_MIDDLE_FINGER,
01418     IBSM_FINGER_POSITION_RIGHT_RING_FINGER,
01419     IBSM_FINGER_POSITION_RIGHT_LITTLE_FINGER,
01420     IBSM_FINGER_POSITION_LEFT_THUMB,
01421     IBSM_FINGER_POSITION_LEFT_INDEX_FINGER,
01422     IBSM_FINGER_POSITION_LEFT_MIDDLE_FINGER,
01423     IBSM_FINGER_POSITION_LEFT_RING_FINGER,
01424     IBSM_FINGER_POSITION_LEFT_LITTLE_FINGER,
01425     IBSM_FINGER_POSITION_PLAIN_RIGHT_FOUR_FINGERS=13,
01426     IBSM_FINGER_POSITION_PLAIN_LEFT_FOUR_FINGERS,
01427     IBSM_FINGER_POSITION_PLAIN_THUMBS,
01428     IBSM_FINGER_POSITION_UNKNOWN_PALM=20,
01429     IBSM_FINGER_POSITION_RIGHT_FULL_PALM,
01430     IBSM_FINGER_POSITION_RIGHT_WRITERS_PALM,
01431     IBSM_FINGER_POSITION_LEFT_FULL_PALM,
01432     IBSM_FINGER_POSITION_LEFT_WRITERS_PALM,
01433     IBSM_FINGER_POSITION_RIGHT_LOWER_PALM,
01434     IBSM_FINGER_POSITION_RIGHT_UPPER_PALM,
01435     IBSM_FINGER_POSITION_LEFT_LOWER_PALM,
01436     IBSM_FINGER_POSITION_LEFT_UPPER_PALM,
01437     IBSM_FINGER_POSITION_RIGHT_OTHER,
01438     IBSM_FINGER_POSITION_LEFT_OTHER,
01439     IBSM_FINGER_POSITION_RIGHT_INTERDIGITAL,
01440     IBSM_FINGER_POSITION_RIGHT_THENAR,
01441     IBSM_FINGER_POSITION_RIGHT_HYPOTHENAR,
01442     IBSM_FINGER_POSITION_LEFT_INTERDIGITAL,
01443     IBSM_FINGER_POSITION_LEFT_THENAR,
01444     IBSM_FINGER_POSITION_LEFT_HYPOTHENAR,
01445     IBSM_FINGER_POSITION_RIGHT_INDEX_AND_MIDDLE=40,
01446     IBSM_FINGER_POSITION_RIGHT_MIDDLE_AND_RING,
01447     IBSM_FINGER_POSITION_RIGHT_RING_AND_LITTLE,
01448     IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE,
01449     IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_RING,
01450     IBSM_FINGER_POSITION_LEFT_RING_AND_LITTLE,
01451     IBSM_FINGER_POSITION_RIGHT_INDEX_AND_LEFT_INDEX,
01452     IBSM_FINGER_POSITION_RIGHT_INDEX_AND_MIDDLE_AND_RING,
01453     IBSM_FINGER_POSITION_RIGHT_MIDDLE_AND_RING_AND_LITTLE,
01454     IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE_AND_RING,
01455     IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_RING_AND_LITTLE,
01456
01457     // HALF PALM
01458     IBSM_FINGER_POSITION_UNKNOWN_HALF_PALM,
01459     IBSM_FINGER_POSITION_RIGHT_HALF_PALM,
01460     IBSM_FINGER_POSITION_LEFT_HALF_PALM,
01461     IBSM_FINGER_POSITION_RIGHT_LOWER_HALF_PALM,
01462     IBSM_FINGER_POSITION_RIGHT_UPPER_HALF_PALM,
01463     IBSM_FINGER_POSITION_LEFT_LOWER_HALF_PALM,
```

```
01464     IBSM_FINGER_POSITION_LEFT_UPPER_HALF_PALM,
01465
01466     IBSM_FINGER_POSITION_LEFT_CENTER_JOINT,
01467     IBSM_FINGER_POSITION_RIGHT_CENTER_JOINT,
01468     IBSM_FINGER_POSITION_LEFT_SIDE_JOINT,
01469     IBSM_FINGER_POSITION_RIGHT_SIDE_JOINT,
01470     IBSM_FINGER_POSITION_LEFT_ROLL_JOINT,
01471     IBSM_FINGER_POSITION_RIGHT_ROLL_JOINT,
01472     IBSM_FINGER_POSITION_LEFT_ROLL_UP,
01473     IBSM_FINGER_POSITION_RIGHT_ROLL_UP
01474 }
```

12.44 Enumeration - Matcher - Capture Device Tech ID

Enumeration of capture device technology IDs to support IBScanMatcher integration.

Enumerations

- enum IBSM_CaptureDeviceTechID {
  IBSM_CAPTURE_DEVICE_UNKNOWN_OR_UNSPECIFIED =0 , IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TIR
  , IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_DIRECT_VIEW_ON_PLATEN , IBSM_CAPTURE_DEVICE_WHITE_L
  ,
  IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TIR , IBSM_CAPTURE_DEVICE_MONOCHROMATIC_
  , IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TOUCHLESS , IBSM_CAPTURE_DEVICE_MONOCH
  ,
  IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_DIRECT_VIEW_ON_PLATEN , IBSM_CAPTURE_DEVICE_M
  , IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TIR , IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_D
  ,
  IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TOUCHLESS , IBSM_CAPTURE_DEVICE_ELECTRO_LUMINESC
  , IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_CAPACITIVE , IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_RF
  ,
  IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_THEMAL , IBSM_CAPTURE_DEVICE_PRESSURE_SENSITIVE
  , IBSM_CAPTURE_DEVICE_ULTRASOUND , IBSM_CAPTURE_DEVICE_MECHANICAL ,
  IBSM_CAPTURE_DEVICE_GLASS_FIBER }

12.44.1 Detailed Description

Enumeration of capture device technology IDs to support IBScanMatcher integration.

12.44.2 Enumeration Type Documentation

12.44.2.1 IBSM_CaptureDeviceTechID

```
enum IBSM_CaptureDeviceTechID
```

Enumerator

IBSM_CAPTURE_DEVICE_UNKNOWN_OR_UNSPECIFIED	
IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TIR	
IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_DIRECT_VIEW_ON_PLATEN	

Enumerator

IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TOUCHLESS	
IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TIR	
IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_DIRECT_VIEW_ON_PLATEN	
IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TOUCHLESS	
IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_TIR	
IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_DIRECT_VIEW_ON_PLATEN	
IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_TOUCHLESS	
IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TIR	
IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_DIRECT_VIEW_ON_PLATEN	
IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TOUCHLESS	
IBSM_CAPTURE_DEVICE_ELECTRO_LUMINESCENT	
IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_CAPACITIVE	
IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_RF	
IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_THEMAL	
IBSM_CAPTURE_DEVICE_PRESSURE_SENSITIVE	
IBSM_CAPTURE_DEVICE_ULTRASOUND	
IBSM_CAPTURE_DEVICE_MECHANICAL	
IBSM_CAPTURE_DEVICE_GLASS_FIBER	

Definition at line 1491 of file IBScanUltimateApi_defs.h.

```
01492 {
01493     IBSM_CAPTURE_DEVICE_UNKNOWN_OR_UNSPECIFIED=0,
01494     IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TIR,
01495     IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01496     IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TOUCHLESS,
01497     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TIR,
01498     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01499     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TOUCHLESS,
01500     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_TIR,
01501     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01502     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_TOUCHLESS,
01503     IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TIR,
01504     IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01505     IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TOUCHLESS,
01506     IBSM_CAPTURE_DEVICE_ELECTRO_LUMINESCENT,
01507     IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_CAPACITIVE,
01508     IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_RF,
01509     IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_THEMAL,
01510     IBSM_CAPTURE_DEVICE_PRESSURE_SENSITIVE,
01511     IBSM_CAPTURE_DEVICE_ULTRASOUND,
01512     IBSM_CAPTURE_DEVICE_MECHANICAL,
01513     IBSM_CAPTURE_DEVICE_GLASS_FIBER
01514 }
```

12.45 Enumeration - Matcher - Supported Device

Enumeration of capture device type IDs to support IBScanMatcher integration.

Enumerations

- enum IBSM_CaptureDeviceTypeID {
  IBSM_CAPTURE_DEVICE_TYPE_ID_UNKNOWN = 0x0000 , IBSM_CAPTURE_DEVICE_TYPE_ID_CURVE = 0x1004 , IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON = 0x1005 , IBSM_CAPTURE_DEVICE_TYPE_ID_SHERLOCK = 0x1010 ,
  IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON_MINI = 0x1020 , IBSM_CAPTURE_DEVICE_TYPE_ID_COLUMBO = 0x1100 , IBSM_CAPTURE_DEVICE_TYPE_ID_HOLMES = 0x1200 , IBSM_CAPTURE_DEVICE_TYPE_ID_KOJAK = 0x1300 ,
  IBSM_CAPTURE_DEVICE_TYPE_ID_FIVE0 = 0x1500 , IBSM_CAPTURE_DEVICE_TYPE_ID_DANNO = 0x1600 , IBSM_CAPTURE_DEVICE_TYPE_ID_MANNIX = 0x1D00 }

12.45.1 Detailed Description

Enumeration of capture device type IDs to support IBScanMatcher integration.

12.45.2 Enumeration Type Documentation

12.45.2.1 IBSM_CaptureDeviceTypeID

```
enum IBSM_CaptureDeviceTypeID
```

Enumerator

IBSM_CAPTURE_DEVICE_TYPE_ID_UNKNOWN	
IBSM_CAPTURE_DEVICE_TYPE_ID_CURVE	
IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON	
IBSM_CAPTURE_DEVICE_TYPE_ID_SHERLOCK	
IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON_MINI	
IBSM_CAPTURE_DEVICE_TYPE_ID_COLUMBO	
IBSM_CAPTURE_DEVICE_TYPE_ID_HOLMES	
IBSM_CAPTURE_DEVICE_TYPE_ID_KOJAK	
IBSM_CAPTURE_DEVICE_TYPE_ID_FIVE0	
IBSM_CAPTURE_DEVICE_TYPE_ID_DANNO	
IBSM_CAPTURE_DEVICE_TYPE_ID_MANNIX	

Definition at line 1531 of file IBScanUltimateApi_defs.h.

```
01532 {
01533     IBSM_CAPTURE_DEVICE_TYPE_ID_UNKNOWN = 0x0000,
01534     IBSM_CAPTURE_DEVICE_TYPE_ID_CURVE = 0x1004,
01535     IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON = 0x1005,
01536     IBSM_CAPTURE_DEVICE_TYPE_ID_SHERLOCK = 0x1010,
01537     IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON_MINI = 0x1020,
01538     IBSM_CAPTURE_DEVICE_TYPE_ID_COLUMBO = 0x1100,
01539     IBSM_CAPTURE_DEVICE_TYPE_ID_HOLMES = 0x1200,
01540     IBSM_CAPTURE_DEVICE_TYPE_ID_KOJAK = 0x1300,
01541     IBSM_CAPTURE_DEVICE_TYPE_ID_FIVE0 = 0x1500,
01542     IBSM_CAPTURE_DEVICE_TYPE_ID_DANNO = 0x1600,
01543     IBSM_CAPTURE_DEVICE_TYPE_ID_MANNIX = 0x1D00,
01544 }
```

12.46 Enumeration - Matcher - Vendor ID

Enumeration of capture device vendor IDs to support IBScanMatcher integration.

Enumerations

- enum IBSM_CaptureDeviceVendorID { IBSM_CAPTURE_DEVICE_VENDOR_ID_UNREPORTED =0x0000 , IBSM_CAPTURE_DEVICE_VENDOR_INTEGRATED_BIOMETRICS =0x113F }

12.46.1 Detailed Description

Enumeration of capture device vendor IDs to support IBScanMatcher integration.

12.46.2 Enumeration Type Documentation

12.46.2.1 IBSM_CaptureDeviceVendorID

```
enum IBSM_CaptureDeviceVendorID
```

Enumerator

IBSM_CAPTURE_DEVICE_VENDOR_ID_UNREPORTED	
IBSM_CAPTURE_DEVICE_VENDOR_INTEGRATED_BIOMETRICS	

Definition at line 1561 of file IBScanUltimateApi_defs.h.

```
01562 {
01563     IBSM_CAPTURE_DEVICE_VENDOR_ID_UNREPORTED=0x0000,
01564     IBSM_CAPTURE_DEVICE_VENDOR_INTEGRATED_BIOMETRICS=0x113F
01565 }
```

12.47 Enumeration - Matcher - Hash Type

Enumeration of the Hashtype.

Enumerations

- enum IBSU_HashType { ENUM_IBSU_HASH_TYPE_SHA256 , ENUM_IBSU_HASH_TYPE_RESERVED }

12.47.1 Detailed Description

Enumeration of the Hashtype.

12.47.2 Enumeration Type Documentation

12.47.2.1 IBSU_HashType

```
enum IBSU_HashType
```

Enumerator

ENUM_IBSU_HASH_TYPE_SHA256	SHA256
ENUM_IBSU_HASH_TYPE_RESERVED	Reserved

Definition at line 1582 of file IBScanUltimateApi_defs.h.

```
01583 {
01585     ENUM_IBSU_HASH_TYPE_SHA256,
01586
01588     ENUM_IBSU_HASH_TYPE_RESERVED
01589 }
```

12.48 Structure - Matcher - Image Data

Container for image information to support IBScanMatcher integration.

Classes

- struct IBSM_ImageData

12.48.1 Detailed Description

Container for image information to support IBScanMatcher integration.

12.49 Enumeration - Matcher - Template Version

Enumeration of Template versions to support ISO/ANSI integration.

Typedefs

- typedef enum enum_IBSM_TemplateVersion IBSM_TemplateVersion

Enumerations

- enum enum_IBSM_TemplateVersion {
  IBSM_TEMPLATE_VERSION_IBISDK_0 =0x00 , IBSM_TEMPLATE_VERSION_IBISDK_1 , IBSM_TEMPLATE_VERSION_IBI
  , IBSM_TEMPLATE_VERSION_IBISDK_3 ,
  IBSM_TEMPLATE_VERSION_NEW_0 =0x10 }

12.49.1 Detailed Description

Enumeration of Template versions to support ISO/ANSI integration.

12.49.2 Typedef Documentation

12.49.2.1 IBSM_TemplateVersion

```
typedef enum enum_IBSM_TemplateVersion IBSM_TemplateVersion
```

12.49.3 Enumeration Type Documentation

12.49.3.1 enum_IBSM_TemplateVersion

```
enum enum_IBSM_TemplateVersion
```

Enumerator

IBSM_TEMPLATE_VERSION_IBISDK↩_0	TEMPLATE_OLD_VERSION
IBSM_TEMPLATE_VERSION_IBISDK↩_1	TEMPLATE_MIX_VERSION
IBSM_TEMPLATE_VERSION_IBISDK↩_2	TEMPLATE_FAST_VERSION
IBSM_TEMPLATE_VERSION_IBISDK↩_3	Secuest 1nd Algorithm
IBSM_TEMPLATE_VERSION_NEW_0	IBSM_NEW_TEMPLATE

Definition at line 1641 of file IBScanUltimateApi_defs.h.

```
01642 {
01644     IBSM_TEMPLATE_VERSION_IBISDK_0=0x00,
01646     IBSM_TEMPLATE_VERSION_IBISDK_1,
01648     IBSM_TEMPLATE_VERSION_IBISDK_2,
01650     IBSM_TEMPLATE_VERSION_IBISDK_3,
01652     IBSM_TEMPLATE_VERSION_NEW_0=0x10
01653 }
```

12.50 Structure - Matcher - Template Data

Container for Template information to support ISO/ANSI integration.

Classes

- struct IBSM_Template

12.50.1 Detailed Description

Container for Template information to support ISO/ANSI integration.

12.51 Enumeration - Matcher - Standard Format Type

Enumeration of Standard Format types to support ISO/ANSI integration.

Enumerations

- enum IBSM_StandardFormat {
  ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2005 , ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2005
  , ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2011 , ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2011
  ,
  ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_378_2004 , ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_381_20
  }

12.51.1 Detailed Description

Enumeration of Standard Format types to support ISO/ANSI integration.

12.51.2 Enumeration Type Documentation

12.51.2.1 IBSM_StandardFormat

```
enum IBSM_StandardFormat
```

Enumerator

ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2005	ISO 19794-2:2005
ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2005	ISO 19794-4:2005
ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2011	ISO 19794-2:2011
ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2011	ISO 19794-4:2011
ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_378_2004	ANSI/INCITS 378:2004
ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_381_2004	ANSI/INCITS 381:2004

Definition at line 1705 of file IBScanUltimateApi_defs.h.

```
01706 {
01708     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2005,
01709
01711     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2005,
01712
01714     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2011,
01715
01717     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2011,
01718
01720     ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_378_2004,
01721
01723     ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_381_2004,
01724 }
```

12.52 Structure - Matcher - Standard Format Data

Container for Standard format data information to support ISO/ANSI integration.

Classes

- struct IBSM_StandardFormatData

12.52.1 Detailed Description

Container for Standard format data information to support ISO/ANSI integration.

12.53 Definition - Callback Interface Functions

Definition for the Interface functions of the Callbacks.

Typedefs

- typedef void(CALLBACK * IBSU_Callback) (const int deviceHandle, void *pContext)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK, called when communication with a device is interrupted.
- typedef void(CALLBACK * IBSU_CallbackPreviewImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE, called when a preview image is available.
- typedef void(CALLBACK * IBSU_CallbackFingerCount) (const int deviceHandle, void *pContext, const IBSU_FingerCountState fingerCountState)

  Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT, called when the finger count changes.
- typedef void(CALLBACK * IBSU_CallbackFingerQuality) (const int deviceHandle, void *pContext, const IBSU_FingerQualityState *pQualityArray, const int qualityArrayCount)

  Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY, called when a finger quality changes.
- typedef void(CALLBACK * IBSU_CallbackDeviceCount) (const int detectedDevices, void *pContext)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT, called when the number of detected devices changes.
- typedef void(CALLBACK * IBSU_CallbackInitProgress) (const int deviceIndex, void *pContext, const int progressValue)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS, called when the initialization progress changes for a device.
- typedef void(CALLBACK * IBSU_CallbackTakingAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION, called for a rolled print acquisition when the rolling should begin.
- typedef void(CALLBACK * IBSU_CallbackCompleteAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION, called for a rolled print acquisition when the rolling capture has completed.
- typedef void(CALLBACK * IBSU_CallbackResultImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image, const IBSU_ImageType imageType, const IBSU_ImageData *pSplitImageArray, const int splitImageArrayCount)

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

- typedef 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)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE_EX, called when the result image is available with extended information.
- typedef void(CALLBACK * IBSU_CallbackClearPlatenAtCapture) (const int deviceHandle, void *pContext, const IBSU_PlatenState platenState)

  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.
- typedef void(CALLBACK * IBSU_CallbackAsyncOpenDevice) (const int deviceIndex, void *pContext, const int deviceHandle, const int errorCode)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE, called when asynchronous device initialization completes.
- typedef void(CALLBACK * IBSU_CallbackNotifyMessage) (const int deviceHandle, void *pContext, const int notifyMessage)

  Callback for ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE, called when a warning message is generated.
- typedef void(CALLBACK * IBSU_CallbackKeyButtons) (const int deviceHandle, void *pContext, const int pressedKeyButtons)

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

12.53.1 Detailed Description

Definition for the Interface functions of the Callbacks.

12.53.2 Typedef Documentation

12.53.2.1 IBSU_Callback

```
typedef void(CALLBACK * IBSU_Callback) (const int deviceHandle, void *pContext)
```

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

IBSU_Callback()

Parameters

deviceHandle	Device handle.
pContext	User context.

Definition at line 1784 of file IBScanUltimateApi_defs.h.

12.53.2.2 IBSU_CallbackAsyncOpenDevice

```
typedef void(CALLBACK * IBSU_CallbackAsyncOpenDevice) (const int deviceIndex, void *pContext,
const int deviceHandle, const int errorCode)
```

Callback for ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE, called when asynchronous device initialization completes.

IBSU_CallbackAsyncOpenDevice()

Parameters

deviceIndex	Zero-based index of device.
pContext	User context.
deviceHandle	Handle for subsequent function calls.
errorCode	Error that prevented initialization from completing.

Definition at line 2054 of file IBScanUltimateApi_defs.h.

12.53.2.3 IBSU_CallbackClearPlatenAtCapture

```
typedef void(CALLBACK * IBSU_CallbackClearPlatenAtCapture) (const int deviceHandle, void *p↩
Context, const IBSU_PlatenState platenState)
```

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.

IBSU_CallbackClearPlatenAtCapture()

Parameters

deviceHandle	Device handle.
pContext	User context.
platenState	Platen state.

Definition at line 2031 of file IBScanUltimateApi_defs.h.

12.53.2.4 IBSU_CallbackCompleteAcquisition

```
typedef void(CALLBACK * IBSU_CallbackCompleteAcquisition) (const int deviceHandle, void *p↩
Context, const IBSU_ImageType imageType)
```

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

IBSU_CallbackCompleteAcquisition()

Parameters

deviceHandle	Device handle.
pContext	User context.
imageType	Type of image being acquired.

Definition at line 1929 of file IBScanUltimateApi_defs.h.

12.53.2.5 IBSU_CallbackDeviceCount

```
typedef void(CALLBACK * IBSU_CallbackDeviceCount) (const int detectedDevices, void *pContext)
```

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

IBSU_CallbackDeviceCount()

Parameters

detectedDevices	Number of detected devices.
pContext	User context.

Definition at line 1867 of file IBScanUltimateApi_defs.h.

12.53.2.6 IBSU_CallbackFingerCount

```
typedef void(CALLBACK * IBSU_CallbackFingerCount) (const int deviceHandle, void *pContext,
const IBSU_FingerCountState fingerCountState)
```

Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT, called when the finger count changes.

IBSU_CallbackFingerCount()

Parameters

deviceHandle	Device handle.
pContext	User context.
fingerCountState	Finger count state.

Definition at line 1825 of file IBScanUltimateApi_defs.h.

12.53.2.7 IBSU_CallbackFingerQuality

```
typedef void(CALLBACK * IBSU_CallbackFingerQuality) (const int deviceHandle, void *pContext,
const IBSU_FingerQualityState *pQualityArray, const int qualityArrayCount)
```

Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY, called when a finger quality changes.

IBSU_CallbackFingerQuality()

Parameters

deviceHandle	Device handle.
pContext	User context.
pQualityArray	Array of finger qualities.
qualityArrayCount	Number of qualities in an array.

Definition at line 1847 of file IBScanUltimateApi_defs.h.

12.53.2.8 IBSU_CallbackInitProgress

```
typedef void(CALLBACK * IBSU_CallbackInitProgress) (const int deviceIndex, void *pContext,
const int progressValue)
```

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

IBSU_CallbackInitProgress()

Parameters

deviceIndex	Zero-based index of a device.
pContext	User context.
progressValue	Initialization progress, as a percent, between 0 and 100, inclusive.

Definition at line 1887 of file IBScanUltimateApi_defs.h.

12.53.2.9 IBSU_CallbackKeyButtons

```
typedef void(CALLBACK * IBSU_CallbackKeyButtons) (const int deviceHandle, void *pContext, const
int pressedKeyButtons)
```

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

IBSU_CallbackKeyButtons()

Parameters

deviceHandle	Device handle.
pContext	User context.
pressedKeyButtons	The key button index that is pressed.

Definition at line 2096 of file IBScanUltimateApi_defs.h.

12.53.2.10 IBSU_CallbackNotifyMessage

```
typedef void(CALLBACK * IBSU_CallbackNotifyMessage) (const int deviceHandle, void *pContext,
const int notifyMessage)
```

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

IBSU_CallbackNotifyMessage()

Parameters

deviceHandle	Device handle.
pContext	User context.
notifyMessage	Status code as defined in IBScanUltimateApi_err.h.

Definition at line 2076 of file IBScanUltimateApi_defs.h.

12.53.2.11 IBSU_CallbackPreviewImage

```
typedef void(CALLBACK * IBSU_CallbackPreviewImage) (const int deviceHandle, void *pContext,
const IBSU_ImageData image)
```

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

IBSU_CallbackPreviewImage()

Parameters

deviceHandle	Device handle.
pContext	User context.
image	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.

Definition at line 1805 of file IBScanUltimateApi_defs.h.

12.53.2.12 IBSU_CallbackResultImage

```
typedef void(CALLBACK * IBSU_CallbackResultImage) (const int deviceHandle, void *pContext,
const IBSU_ImageData image, const IBSU_ImageType imageType, const IBSU_ImageData *pSplitImage↩
Array, const int splitImageArrayCount)
```

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

IBSU_CallbackResultImage()

Parameters

deviceHandle	Device handle.
pContext	User context.
image	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	Image type.
pSplitImageArray	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	Number of finger images split from result images.

Definition at line 1961 of file IBScanUltimateApi_defs.h.

12.53.2.13 IBSU_CallbackResultImageEx

```
typedef 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 *pSegmentImage↩
Array, const IBSU_SegmentPosition *pSegmentPositionArray)
```

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

IBSU_CallbackResultImageEx()

Parameters

deviceHandle	Device handle.
pContext	User context.
imageStatus	Status from result image acquisition. See error codes in 'IBScanUltimateApi_err'.
image	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	Image type.
detectedFingerCount	Number of detected fingers.
segmentImageArrayCount	Number of finger images split from result images.
pSegmentImageArray	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	Array of structures with position data for individual fingers split from result image.

Definition at line 2004 of file IBScanUltimateApi_defs.h.

12.53.2.14 IBSU_CallbackTakingAcquisition

```
typedef void(CALLBACK * IBSU_CallbackTakingAcquisition) (const int deviceHandle, void *p↩
Context, const IBSU_ImageType imageType)
```

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

IBSU_CallbackTakingAcquisition()

Parameters

deviceHandle	Device handle.
pContext	User context.
imageType	Type of image being acquired.

Definition at line 1908 of file IBScanUltimateApi_defs.h.

12.54 Definition - Error Code - General

Error Code from 0 to -11.

Macros

- #define IBSU_STATUS_OK 0
- #define IBSU_ERR_INVALID_PARAM_VALUE -1
- #define IBSU_ERR_MEM_ALLOC -2
- #define IBSU_ERR_NOT_SUPPORTED -3
- #define IBSU_ERR_FILE_OPEN -4
- #define IBSU_ERR_FILE_READ -5
- #define IBSU_ERR_RESOURCE_LOCKED -6
- #define IBSU_ERR_MISSING_RESOURCE -7
- #define IBSU_ERR_INVALID_ACCESS_POINTER -8
- #define IBSU_ERR_THREAD_CREATE -9
- #define IBSU_ERR_COMMAND_FAILED -10
- #define IBSU_ERR_LIBRARY_UNLOAD_FAILED -11

12.54.1 Detailed Description

Error Code from 0 to -11.

12.54.2 Macro Definition Documentation

12.54.2.1 IBSU_ERR_COMMAND_FAILED

```
#define IBSU_ERR_COMMAND_FAILED -10
```

Generic command execution failed.

Definition at line 115 of file IBScanUltimateApi_err.h.

12.54.2.2 IBSU_ERR_FILE_OPEN

```
#define IBSU_ERR_FILE_OPEN -4
```

File (USB handle, pipe, or image file) open failed.

Definition at line 103 of file IBScanUltimateApi_err.h.

12.54.2.3 IBSU_ERR_FILE_READ

```
#define IBSU_ERR_FILE_READ -5
```

File (USB handle, pipe, or image file) read failed.

Definition at line 105 of file IBScanUltimateApi_err.h.

12.54.2.4 IBSU_ERR_INVALID_ACCESS_POINTER

```
#define IBSU_ERR_INVALID_ACCESS_POINTER -8
```

Invalid access pointer address.

Definition at line 111 of file IBScanUltimateApi_err.h.

12.54.2.5 IBSU_ERR_INVALID_PARAM_VALUE

```
#define IBSU_ERR_INVALID_PARAM_VALUE -1
```

Invalid parameter value.

Definition at line 97 of file IBScanUltimateApi_err.h.

12.54.2.6 IBSU_ERR_LIBRARY_UNLOAD_FAILED

```
#define IBSU_ERR_LIBRARY_UNLOAD_FAILED -11
```

The library unload failed.

Definition at line 117 of file IBScanUltimateApi_err.h.

12.54.2.7 IBSU_ERR_MEM_ALLOC

```
#define IBSU_ERR_MEM_ALLOC -2
```

Insufficient memory.

Definition at line 99 of file IBScanUltimateApi_err.h.

12.54.2.8 IBSU_ERR_MISSING_RESOURCE

```
#define IBSU_ERR_MISSING_RESOURCE -7
```

Failure due to a missing resource (e.g. DLL file).

Definition at line 109 of file IBScanUltimateApi_err.h.

12.54.2.9 IBSU_ERR_NOT_SUPPORTED

```
#define IBSU_ERR_NOT_SUPPORTED -3
```

Requested functionality isn't supported.

Definition at line 101 of file IBScanUltimateApi_err.h.

12.54.2.10 IBSU_ERR_RESOURCE_LOCKED

```
#define IBSU_ERR_RESOURCE_LOCKED -6
```

Failure due to a locked resource.

Definition at line 107 of file IBScanUltimateApi_err.h.

12.54.2.11 IBSU_ERR_THREAD_CREATE

```
#define IBSU_ERR_THREAD_CREATE -9
```

Thread creation failed.

Definition at line 113 of file IBScanUltimateApi_err.h.

12.54.2.12 IBSU_STATUS_OK

```
#define IBSU_STATUS_OK 0
```

Function completed successfully.

Definition at line 95 of file IBScanUltimateApi_err.h.

12.55 Definition - Error Code - Low Level I/O

Error Code from -100 to -107.

Macros

- #define IBSU_ERR_CHANNEL_IO_COMMAND_FAILED -100
- #define IBSU_ERR_CHANNEL_IO_READ_FAILED -101
- #define IBSU_ERR_CHANNEL_IO_WRITE_FAILED -102
- #define IBSU_ERR_CHANNEL_IO_READ_TIMEOUT -103
- #define IBSU_ERR_CHANNEL_IO_WRITE_TIMEOUT -104
- #define IBSU_ERR_CHANNEL_IO_UNEXPECTED_FAILED -105
- #define IBSU_ERR_CHANNEL_IO_INVALID_HANDLE -106
- #define IBSU_ERR_CHANNEL_IO_WRONG_PIPE_INDEX -107

12.55.1 Detailed Description

Error Code from -100 to -107.

12.55.2 Macro Definition Documentation

12.55.2.1 IBSU_ERR_CHANNEL_IO_COMMAND_FAILED

```
#define IBSU_ERR_CHANNEL_IO_COMMAND_FAILED -100
```

Command execution failed.

Definition at line 132 of file IBScanUltimateApi_err.h.

12.55.2.2 IBSU_ERR_CHANNEL_IO_INVALID_HANDLE

```
#define IBSU_ERR_CHANNEL_IO_INVALID_HANDLE -106
```

I/O handle state is invalid; reinitialization (close then open) required.

Definition at line 144 of file IBScanUltimateApi_err.h.

12.55.2.3 IBSU_ERR_CHANNEL_IO_READ_FAILED

```
#define IBSU_ERR_CHANNEL_IO_READ_FAILED -101
```

Input communication failed.

Definition at line 134 of file IBScanUltimateApi_err.h.

12.55.2.4 IBSU_ERR_CHANNEL_IO_READ_TIMEOUT

```
#define IBSU_ERR_CHANNEL_IO_READ_TIMEOUT -103
```

Input command execution timed out, but device communication is alive.

Definition at line 138 of file IBScanUltimateApi_err.h.

12.55.2.5 IBSU_ERR_CHANNEL_IO_UNEXPECTED_FAILED

```
#define IBSU_ERR_CHANNEL_IO_UNEXPECTED_FAILED -105
```

Unexpected communication failed. (Only used on IBTraceLogger.)

Definition at line 142 of file IBScanUltimateApi_err.h.

12.55.2.6 IBSU_ERR_CHANNEL_IO_WRITE_FAILED

```
#define IBSU_ERR_CHANNEL_IO_WRITE_FAILED -102
```

Output communication failed.

Definition at line 136 of file IBScanUltimateApi_err.h.

12.55.2.7 IBSU_ERR_CHANNEL_IO_WRITE_TIMEOUT

```
#define IBSU_ERR_CHANNEL_IO_WRITE_TIMEOUT -104
```

Output command execution timed out, but device communication is alive.

Definition at line 140 of file IBScanUltimateApi_err.h.

12.55.2.8 IBSU_ERR_CHANNEL_IO_WRONG_PIPE_INDEX

```
#define IBSU_ERR_CHANNEL_IO_WRONG_PIPE_INDEX -107
```

I/O pipe index is invalid; reinitialization (close then open) required.

Definition at line 146 of file IBScanUltimateApi_err.h.

12.56 Definition - Error Code - Device Related

Error Code from -200 to -222.

Macros

- #define IBSU_ERR_DEVICE_IO -200
- #define IBSU_ERR_DEVICE_NOT_FOUND -201
- #define IBSU_ERR_DEVICE_NOT_MATCHED -202
- #define IBSU_ERR_DEVICE_ACTIVE -203
- #define IBSU_ERR_DEVICE_NOT_INITIALIZED -204
- #define IBSU_ERR_DEVICE_INVALID_STATE -205
- #define IBSU_ERR_DEVICE_BUSY -206
- #define IBSU_ERR_DEVICE_NOT_SUPPORTED_FEATURE -207
- #define IBSU_ERR_INVALID_LICENSE -208
- #define IBSU_ERR_USB20_REQUIRED -209
- #define IBSU_ERR_DEVICE_ENABLED_POWER_SAVE_MODE -210
- #define IBSU_ERR_DEVICE_NEED_UPDATE_FIRMWARE -211
- #define IBSU_ERR_DEVICE_NEED_CALIBRATE_TOF -212
- #define IBSU_ERR_DEVICE_INVALID_CALIBRATION_DATA -213
- #define IBSU_ERR_DEVICE_HIGHER_SDK_REQUIRED -214
- #define IBSU_ERR_DEVICE_LOCK_INVALID_BUFF -215
- #define IBSU_ERR_DEVICE_LOCK_INFO_EMPTY -216
- #define IBSU_ERR_DEVICE_LOCK_INFO_NOT_MATCHED -217
- #define IBSU_ERR_DEVICE_LOCK_INVALID_CHECKSUM -218
- #define IBSU_ERR_DEVICE_LOCK_INVALID_KEY -219
- #define IBSU_ERR_DEVICE_LOCK_LOCKED -220
- #define IBSU_ERR_DEVICE_LOCK_ILLEGAL_DEVICE -221
- #define IBSU_ERR_DEVICE_LOCK_INVALID_SERIAL_FORMAT -222

12.56.1 Detailed Description

Error Code from -200 to -222.

12.56.2 Macro Definition Documentation

12.56.2.1 IBSU_ERR_DEVICE_ACTIVE

```
#define IBSU_ERR_DEVICE_ACTIVE -203
```

Initialization failed because it is in use by another thread/process.

Definition at line 168 of file IBScanUltimateApi_err.h.

12.56.2.2 IBSU_ERR_DEVICE_BUSY

```
#define IBSU_ERR_DEVICE_BUSY -206
```

Another thread is currently using device functions.

Definition at line 174 of file IBScanUltimateApi_err.h.

12.56.2.3 IBSU_ERR_DEVICE_ENABLED_POWER_SAVE_MODE

```
#define IBSU_ERR_DEVICE_ENABLED_POWER_SAVE_MODE -210
```

Device has enabled the power save mode.

Definition at line 182 of file IBScanUltimateApi_err.h.

12.56.2.4 IBSU_ERR_DEVICE_HIGHER_SDK_REQUIRED

```
#define IBSU_ERR_DEVICE_HIGHER_SDK_REQUIRED -214
```

Device is required to connect higher SDK version for running

Definition at line 190 of file IBScanUltimateApi_err.h.

12.56.2.5 IBSU_ERR_DEVICE_INVALID_CALIBRATION_DATA

```
#define IBSU_ERR_DEVICE_INVALID_CALIBRATION_DATA -213
```

Invalid calibration data from the device.

Definition at line 188 of file IBScanUltimateApi_err.h.

12.56.2.6 IBSU_ERR_DEVICE_INVALID_STATE

```
#define IBSU_ERR_DEVICE_INVALID_STATE -205
```

Device state is invalid; reinitialization (exit then initialization) required.

Definition at line 172 of file IBScanUltimateApi_err.h.

12.56.2.7 IBSU_ERR_DEVICE_IO

```
#define IBSU_ERR_DEVICE_IO -200
```

Device communication failed.

Definition at line 162 of file IBScanUltimateApi_err.h.

12.56.2.8 IBSU_ERR_DEVICE_LOCK_ILLEGAL_DEVICE

```
#define IBSU_ERR_DEVICE_LOCK_ILLEGAL_DEVICE -221
```

The device is not valid from the license file

Definition at line 204 of file IBScanUltimateApi_err.h.

12.56.2.9 IBSU_ERR_DEVICE_LOCK_INFO_EMPTY

```
#define IBSU_ERR_DEVICE_LOCK_INFO_EMPTY -216
```

The Lock-info Buff is empty.

Definition at line 194 of file IBScanUltimateApi_err.h.

12.56.2.10 IBSU_ERR_DEVICE_LOCK_INFO_NOT_MATCHED

```
#define IBSU_ERR_DEVICE_LOCK_INFO_NOT_MATCHED -217
```

The Customer Key to the devices is not registered.

Definition at line 196 of file IBScanUltimateApi_err.h.

12.56.2.11 IBSU_ERR_DEVICE_LOCK_INVALID_BUFF

```
#define IBSU_ERR_DEVICE_LOCK_INVALID_BUFF -215
```

The Lock-info Buff is not valid.

Definition at line 192 of file IBScanUltimateApi_err.h.

12.56.2.12 IBSU_ERR_DEVICE_LOCK_INVALID_CHECKSUM

```
#define IBSU_ERR_DEVICE_LOCK_INVALID_CHECKSUM -218
```

Checksums between buffer and calculated are different.

Definition at line 198 of file IBScanUltimateApi_err.h.

12.56.2.13 IBSU_ERR_DEVICE_LOCK_INVALID_KEY

```
#define IBSU_ERR_DEVICE_LOCK_INVALID_KEY -219
```

When Customer key is invalid.

Definition at line 200 of file IBScanUltimateApi_err.h.

12.56.2.14 IBSU_ERR_DEVICE_LOCK_INVALID_SERIAL_FORMAT

```
#define IBSU_ERR_DEVICE_LOCK_INVALID_SERIAL_FORMAT -222
```

The serial number format is not valid

Definition at line 206 of file IBScanUltimateApi_err.h.

12.56.2.15 IBSU_ERR_DEVICE_LOCK_LOCKED

```
#define IBSU_ERR_DEVICE_LOCK_LOCKED -220
```

The device is locked.

Definition at line 202 of file IBScanUltimateApi_err.h.

12.56.2.16 IBSU_ERR_DEVICE_NEED_CALIBRATE_TOF

```
#define IBSU_ERR_DEVICE_NEED_CALIBRATE_TOF -212
```

Need to calibrate TOF.

Definition at line 186 of file IBScanUltimateApi_err.h.

12.56.2.17 IBSU_ERR_DEVICE_NEED_UPDATE_FIRMWARE

```
#define IBSU_ERR_DEVICE_NEED_UPDATE_FIRMWARE -211
```

Need to update firmware.

Definition at line 184 of file IBScanUltimateApi_err.h.

12.56.2.18 IBSU_ERR_DEVICE_NOT_FOUND

```
#define IBSU_ERR_DEVICE_NOT_FOUND -201
```

No device is detected/active.

Definition at line 164 of file IBScanUltimateApi_err.h.

12.56.2.19 IBSU_ERR_DEVICE_NOT_INITIALIZED

```
#define IBSU_ERR_DEVICE_NOT_INITIALIZED -204
```

Device needs to be initialized.

Definition at line 170 of file IBScanUltimateApi_err.h.

12.56.2.20 IBSU_ERR_DEVICE_NOT_MATCHED

```
#define IBSU_ERR_DEVICE_NOT_MATCHED -202
```

No matching device is detected.

Definition at line 166 of file IBScanUltimateApi_err.h.

12.56.2.21 IBSU_ERR_DEVICE_NOT_SUPPORTED_FEATURE

```
#define IBSU_ERR_DEVICE_NOT_SUPPORTED_FEATURE -207
```

No hardware support for requested function.

Definition at line 176 of file IBScanUltimateApi_err.h.

12.56.2.22 IBSU_ERR_INVALID_LICENSE

```
#define IBSU_ERR_INVALID_LICENSE -208
```

The license is invalid or does not match the device.

Definition at line 178 of file IBScanUltimateApi_err.h.

12.56.2.23 IBSU_ERR_USB20_REQUIRED

```
#define IBSU_ERR_USB20_REQUIRED -209
```

Device is connected to a full-speed USB port but high-speed is required.

Definition at line 180 of file IBScanUltimateApi_err.h.

12.57 Definition - Error Code - Image Capture Related

Error Code from -300 to -308.

Macros

- #define IBSU_ERR_CAPTURE_COMMAND_FAILED -300
- #define IBSU_ERR_CAPTURE_STOP -301
- #define IBSU_ERR_CAPTURE_TIMEOUT -302
- #define IBSU_ERR_CAPTURE_STILL_RUNNING -303
- #define IBSU_ERR_CAPTURE_NOT_RUNNING -304
- #define IBSU_ERR_CAPTURE_INVALID_MODE -305
- #define IBSU_ERR_CAPTURE_ALGORITHM -306
- #define IBSU_ERR_CAPTURE_ROLLING -307
- #define IBSU_ERR_CAPTURE_ROLLING_TIMEOUT -308

12.57.1 Detailed Description

Error Code from -300 to -308.

12.57.2 Macro Definition Documentation

12.57.2.1 IBSU_ERR_CAPTURE_ALGORITHM

```
#define IBSU_ERR_CAPTURE_ALGORITHM -306
```

Generic algorithm processing failure.

Definition at line 233 of file IBScanUltimateApi_err.h.

12.57.2.2 IBSU_ERR_CAPTURE_COMMAND_FAILED

```
#define IBSU_ERR_CAPTURE_COMMAND_FAILED -300
```

Image acquisition failed.

Definition at line 221 of file IBScanUltimateApi_err.h.

12.57.2.3 IBSU_ERR_CAPTURE_INVALID_MODE

```
#define IBSU_ERR_CAPTURE_INVALID_MODE -305
```

Capture mode is not valid or not supported.

Definition at line 231 of file IBScanUltimateApi_err.h.

12.57.2.4 IBSU_ERR_CAPTURE_NOT_RUNNING

```
#define IBSU_ERR_CAPTURE_NOT_RUNNING -304
```

A capture is not running.

Definition at line 229 of file IBScanUltimateApi_err.h.

12.57.2.5 IBSU_ERR_CAPTURE_ROLLING

```
#define IBSU_ERR_CAPTURE_ROLLING -307
```

Image processing failure at rolled finger print processing.

Definition at line 235 of file IBScanUltimateApi_err.h.

12.57.2.6 IBSU_ERR_CAPTURE_ROLLING_TIMEOUT

```
#define IBSU_ERR_CAPTURE_ROLLING_TIMEOUT -308
```

No roll start detected within a defined timeout period.

Definition at line 237 of file IBScanUltimateApi_err.h.

12.57.2.7 IBSU_ERR_CAPTURE_STILL_RUNNING

```
#define IBSU_ERR_CAPTURE_STILL_RUNNING -303
```

A capture is still running.

Definition at line 227 of file IBScanUltimateApi_err.h.

12.57.2.8 IBSU_ERR_CAPTURE_STOP

```
#define IBSU_ERR_CAPTURE_STOP -301
```

Stop capture failed.

Definition at line 223 of file IBScanUltimateApi_err.h.

12.57.2.9 IBSU_ERR_CAPTURE_TIMEOUT

```
#define IBSU_ERR_CAPTURE_TIMEOUT -302
```

Timeout during capturing.

Definition at line 225 of file IBScanUltimateApi_err.h.

12.58 Definition - Error Code - Client Window Related

Error Code from -400 to -402.

Macros

- #define IBSU_ERR_CLIENT_WINDOW -400
- #define IBSU_ERR_CLIENT_WINDOW_NOT_CREATE -401
- #define IBSU_ERR_INVALID_OVERLAY_HANDLE -402

12.58.1 Detailed Description

Error Code from -400 to -402.

12.58.2 Macro Definition Documentation

12.58.2.1 IBSU_ERR_CLIENT_WINDOW

```
#define IBSU_ERR_CLIENT_WINDOW -400
```

Generic client window failure.

Definition at line 252 of file IBScanUltimateApi_err.h.

12.58.2.2 IBSU_ERR_CLIENT_WINDOW_NOT_CREATE

```
#define IBSU_ERR_CLIENT_WINDOW_NOT_CREATE -401
```

No client window has been created.

Definition at line 254 of file IBScanUltimateApi_err.h.

12.58.2.3 IBSU_ERR_INVALID_OVERLAY_HANDLE

```
#define IBSU_ERR_INVALID_OVERLAY_HANDLE -402
```

Invalid overlay handle.

Definition at line 256 of file IBScanUltimateApi_err.h.

12.59 Definition - Error Code - NBIS Related

Error Code from -500 to -504.

Macros

- #define IBSU_ERR_NBIS_NFIQ_FAILED -500
- #define IBSU_ERR_NBIS_WSQ_ENCODE_FAILED -501
- #define IBSU_ERR_NBIS_WSQ_DECODE_FAILED -502
- #define IBSU_ERR_NBIS_PNG_ENCODE_FAILED -503
- #define IBSU_ERR_NBIS_JP2_ENCODE_FAILED -504

12.59.1 Detailed Description

Error Code from -500 to -504.

12.59.2 Macro Definition Documentation

12.59.2.1 IBSU_ERR_NBIS_JP2_ENCODE_FAILED

```
#define IBSU_ERR_NBIS_JP2_ENCODE_FAILED -504
```

JP2 encode failed.

Definition at line 279 of file IBScanUltimateApi_err.h.

12.59.2.2 IBSU_ERR_NBIS_NFIQ_FAILED

```
#define IBSU_ERR_NBIS_NFIQ_FAILED -500
```

Getting NFIQ score failed.

Definition at line 271 of file IBScanUltimateApi_err.h.

12.59.2.3 IBSU_ERR_NBIS_PNG_ENCODE_FAILED

```
#define IBSU_ERR_NBIS_PNG_ENCODE_FAILED -503
```

PNG encode failed.

Definition at line 277 of file IBScanUltimateApi_err.h.

12.59.2.4 IBSU_ERR_NBIS_WSQ_DECODE_FAILED

```
#define IBSU_ERR_NBIS_WSQ_DECODE_FAILED -502
```

WSQ decode failed.

Definition at line 275 of file IBScanUltimateApi_err.h.

12.59.2.5 IBSU_ERR_NBIS_WSQ_ENCODE_FAILED

```
#define IBSU_ERR_NBIS_WSQ_ENCODE_FAILED -501
```

WSQ encode failed.

Definition at line 273 of file IBScanUltimateApi_err.h.

12.60 Definition - Error Code - Matcher Related

Error Code from -600 to -603.

Macros

- #define IBSU_ERR_DUPLICATE_EXTRACTION_FAILED -600
- #define IBSU_ERR_DUPLICATE_ALREADY_USED -601
- #define IBSU_ERR_DUPLICATE_SEGMENTATION_FAILED -602
- #define IBSU_ERR_DUPLICATE_MATCHING_FAILED -603

12.60.1 Detailed Description

Error Code from -600 to -603.

12.60.2 Macro Definition Documentation

12.60.2.1 IBSU_ERR_DUPLICATE_ALREADY_USED

```
#define IBSU_ERR_DUPLICATE_ALREADY_USED -601
```

The image of the fingerposition is already in use. in IBSU_ADDFingerImage

Definition at line 297 of file IBScanUltimateApi_err.h.

12.60.2.2 IBSU_ERR_DUPLICATE_EXTRACTION_FAILED

```
#define IBSU_ERR_DUPLICATE_EXTRACTION_FAILED -600
```

The extraction from the fingerimage failed in IBSU_ADDFingerImage and DLL_IsFingerDuplicated

Definition at line 295 of file IBScanUltimateApi_err.h.

12.60.2.3 IBSU_ERR_DUPLICATE_MATCHING_FAILED

```
#define IBSU_ERR_DUPLICATE_MATCHING_FAILED -603
```

Found small extractions in IBSM_MatchingTemplate

Definition at line 301 of file IBScanUltimateApi_err.h.

12.60.2.4 IBSU_ERR_DUPLICATE_SEGMENTATION_FAILED

```
#define IBSU_ERR_DUPLICATE_SEGMENTATION_FAILED -602
```

Found segment fingercounts are not two or more in IBSU_IsValidFingerGeometry

Definition at line 299 of file IBScanUltimateApi_err.h.

12.61 Definition - Error Code - PAD Related

Error Code -700.

Macros

- #define IBSU_ERR_PAD_PROPERTY_DISABLED -700

12.61.1 Detailed Description

Error Code -700.

12.61.2 Macro Definition Documentation

12.61.2.1 IBSU_ERR_PAD_PROPERTY_DISABLED

```
#define IBSU_ERR_PAD_PROPERTY_DISABLED -700
```

PAD Property is not enabled.

Definition at line 316 of file IBScanUltimateApi_err.h.

12.62 Definition - Error Code - ISO/ANSI

Error Code -800.

Macros

- #define IBSU_ERR_INCORRECT_STANDARD_FORMAT -800

12.62.1 Detailed Description

Error Code -800.

12.62.2 Macro Definition Documentation

12.62.2.1 IBSU_ERR_INCORRECT_STANDARD_FORMAT

```
#define IBSU_ERR_INCORRECT_STANDARD_FORMAT -800
```

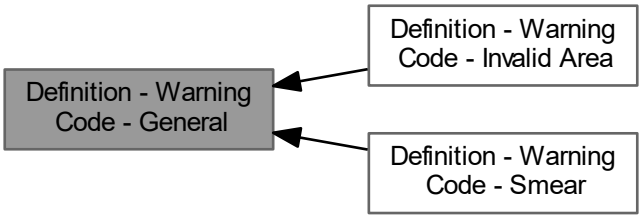
Standard data(ISO or ANSI) is incorrect.

Definition at line 331 of file IBScanUltimateApi_err.h.

12.63 Definition - Warning Code - General

General Warning Codes.

Collaboration diagram for Definition - Warning Code - General:



Modules

- Definition - Warning Code - Smear

  Warning Codes related to Smear detection from roll-finger image.
- Definition - Warning Code - Invalid Area

  Warning Codes related to invalid area which occurs when user put fingers on the area close to the edge of the active area.

Macros

- #define IBSU_WRN_CHANNEL_IO_FRAME_MISSING 100
- #define IBSU_WRN_CHANNEL_IO_CAMERA_WRONG 101
- #define IBSU_WRN_CHANNEL_IO_SLEEP_STATUS 102
- #define IBSU_WRN_OUTDATED_FIRMWARE 200
- #define IBSU_WRN_ALREADY_INITIALIZED 201
- #define IBSU_WRN_API_DEPRECATED 202
- #define IBSU_WRN_ALREADY_ENHANCED_IMAGE 203
- #define IBSU_WRN_BGET_IMAGE 300
- #define IBSU_WRN_ROLLING_NOT_RUNNING 301
- #define IBSU_WRN_NO_FINGER 302
- #define IBSU_WRN_INCORRECT_FINGERS 303
- #define IBSU_WRN_EMPTY_IBSM_RESULT_IMAGE 400
- #define IBSU_WRN_INVALID_BRIGHTNESS_FINGERS 600
- #define IBSU_WRN_WET_FINGERS 601
- #define IBSU_WRN_MULTIPLE_FINGERS_DURING_ROLL 602
- #define IBSU_WRN_SPOOF_DETECTED 603
- #define IBSU_WRN_ROLLING_SLIP_DETECTED 604
- #define IBSU_WRN_SPOOF_INIT_FAILED 605
- #define IBSU_WRN_MATCHER_NO_MATCH 700
- #define IBSU_WRN_MATCHER_ALREADY_REGISTERED 701

12.63.1 Detailed Description

General Warning Codes.

12.63.2 Macro Definition Documentation

12.63.2.1 IBSU_WRN_ALREADY_ENHANCED_IMAGE

```
#define IBSU_WRN_ALREADY_ENHANCED_IMAGE 203
```

Image is already enhanced.

Definition at line 359 of file IBScanUltimateApi_err.h.

12.63.2.2 IBSU_WRN_ALREADY_INITIALIZED

```
#define IBSU_WRN_ALREADY_INITIALIZED 201
```

Device/component has already been initialized and is ready to be used.

Definition at line 355 of file IBScanUltimateApi_err.h.

12.63.2.3 IBSU_WRN_API_DEPRECATED

```
#define IBSU_WRN_API_DEPRECATED 202
```

API function was deprecated.

Definition at line 357 of file IBScanUltimateApi_err.h.

12.63.2.4 IBSU_WRN_BGET_IMAGE

```
#define IBSU_WRN_BGET_IMAGE 300
```

Device still has not received the first image frame.

Definition at line 361 of file IBScanUltimateApi_err.h.

12.63.2.5 IBSU_WRN_CHANNEL_IO_CAMERA_WRONG

```
#define IBSU_WRN_CHANNEL_IO_CAMERA_WRONG 101
```

Camera work is wrong. reset is requied (Only used on IBTraceLogger.)

Definition at line 349 of file IBScanUltimateApi_err.h.

12.63.2.6 IBSU_WRN_CHANNEL_IO_FRAME_MISSING

```
#define IBSU_WRN_CHANNEL_IO_FRAME_MISSING 100
```

Missing an image frame. (Only used on IBTraceLogger.)

Definition at line 347 of file IBScanUltimateApi_err.h.

12.63.2.7 IBSU_WRN_CHANNEL_IO_SLEEP_STATUS

```
#define IBSU_WRN_CHANNEL_IO_SLEEP_STATUS 102
```

Camera work is wrong. IO Write failure or IO Read failure

Definition at line 351 of file IBScanUltimateApi_err.h.

12.63.2.8 IBSU_WRN_EMPTY_IBSM_RESULT_IMAGE

```
#define IBSU_WRN_EMPTY_IBSM_RESULT_IMAGE 400
```

Empty result image.

Definition at line 369 of file IBScanUltimateApi_err.h.

12.63.2.9 IBSU_WRN_INCORRECT_FINGERS

```
#define IBSU_WRN_INCORRECT_FINGERS 303
```

Incorrect fingers detected in result image.

Definition at line 367 of file IBScanUltimateApi_err.h.

12.63.2.10 IBSU_WRN_INVALID_BRIGHTNESS_FINGERS

```
#define IBSU_WRN_INVALID_BRIGHTNESS_FINGERS 600
```

When a finger doesn't meet image brightness criteria

Definition at line 371 of file IBScanUltimateApi_err.h.

12.63.2.11 IBSU_WRN_MATCHER_ALREADY_REGISTERED

```
#define IBSU_WRN_MATCHER_ALREADY_REGISTERED 701
```

Definition at line 385 of file IBScanUltimateApi_err.h.

12.63.2.12 IBSU_WRN_MATCHER_NO_MATCH

```
#define IBSU_WRN_MATCHER_NO_MATCH 700
```

Definition at line 383 of file IBScanUltimateApi_err.h.

12.63.2.13 IBSU_WRN_MULTIPLE_FINGERS_DURING_ROLL

```
#define IBSU_WRN_MULTIPLE_FINGERS_DURING_ROLL 602
```

Detected multiple fingers during roll

Definition at line 375 of file IBScanUltimateApi_err.h.

12.63.2.14 IBSU_WRN_NO_FINGER

```
#define IBSU_WRN_NO_FINGER 302
```

No finger detected in result image.

Definition at line 365 of file IBScanUltimateApi_err.h.

12.63.2.15 IBSU_WRN_OUTDATED_FIRMWARE

```
#define IBSU_WRN_OUTDATED_FIRMWARE 200
```

Device firmware version outdated.

Definition at line 353 of file IBScanUltimateApi_err.h.

12.63.2.16 IBSU_WRN_ROLLING_NOT_RUNNING

```
#define IBSU_WRN_ROLLING_NOT_RUNNING 301
```

Rolling has not started.

Definition at line 363 of file IBScanUltimateApi_err.h.

12.63.2.17 IBSU_WRN_ROLLING_SLIP_DETECTED

```
#define IBSU_WRN_ROLLING_SLIP_DETECTED 604
```

Detected slip finger

Definition at line 379 of file IBScanUltimateApi_err.h.

12.63.2.18 IBSU_WRN_SPOOF_DETECTED

```
#define IBSU_WRN_SPOOF_DETECTED 603
```

Detected spoof finger

Definition at line 377 of file IBScanUltimateApi_err.h.

12.63.2.19 IBSU_WRN_SPOOF_INIT_FAILED

```
#define IBSU_WRN_SPOOF_INIT_FAILED 605
```

Spoof initalize failed

Definition at line 381 of file IBScanUltimateApi_err.h.

12.63.2.20 IBSU_WRN_WET_FINGERS

```
#define IBSU_WRN_WET_FINGERS 601
```

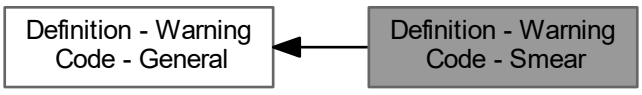
Wet finger detected

Definition at line 373 of file IBScanUltimateApi_err.h.

12.64 Definition - Warning Code - Smear

Warning Codes related to Smear detection from roll-finger image.

Collaboration diagram for Definition - Warning Code - Smear:



Macros

- #define IBSU_WRN_ROLLING_SMEAR 304
- #define IBSU_WRN_ROLLING_SHIFTED_HORIZONTALLY (IBSU_WRN_ROLLING_SMEAR | 1)
- #define IBSU_WRN_ROLLING_SHIFTED_VERTICALLY (IBSU_WRN_ROLLING_SMEAR | 2)

12.64.1 Detailed Description

Warning Codes related to Smear detection from roll-finger image.

12.64.2 Macro Definition Documentation

12.64.2.1 IBSU_WRN_ROLLING_SHIFTED_HORIZONTALLY

```
#define IBSU_WRN_ROLLING_SHIFTED_HORIZONTALLY (IBSU_WRN_ROLLING_SMEAR | 1)
```

Rolled finger was shifted horizontally.

Definition at line 403 of file IBScanUltimateApi_err.h.

12.64.2.2 IBSU_WRN_ROLLING_SHIFTED_VERTICALLY

```
#define IBSU_WRN_ROLLING_SHIFTED_VERTICALLY (IBSU_WRN_ROLLING_SMEAR | 2)
```

Rolled finger was shifted vertically.

Definition at line 405 of file IBScanUltimateApi_err.h.

12.64.2.3 IBSU_WRN_ROLLING_SMEAR

```
#define IBSU_WRN_ROLLING_SMEAR 304
```

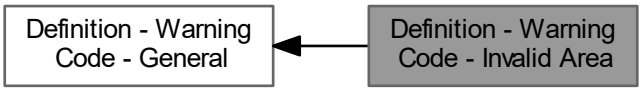
Smear detected in rolled result image.

Definition at line 401 of file IBScanUltimateApi_err.h.

12.65 Definition - Warning Code - Invalid Area

Warning Codes related to invalid area which occurs when user put fingers on the area close to the edge of the active area.

Collaboration diagram for Definition - Warning Code - Invalid Area:



Macros

- #define IBSU_WRN_QUALITY_INVALID_AREA 512
- #define IBSU_WRN_QUALITY_INVALID_AREA_HORIZONTALLY (IBSU_WRN_QUALITY_INVALID_AREA | 1)
- #define IBSU_WRN_QUALITY_INVALID_AREA_VERTICALLY (IBSU_WRN_QUALITY_INVALID_AREA | 2)

12.65.1 Detailed Description

Warning Codes related to invalid area which occurs when user put fingers on the area close to the edge of the active area.

12.65.2 Macro Definition Documentation

12.65.2.1 IBSU_WRN_QUALITY_INVALID_AREA

```
#define IBSU_WRN_QUALITY_INVALID_AREA 512
```

When a finger is located in the invalid area

Definition at line 421 of file IBScanUltimateApi_err.h.

12.65.2.2 IBSU_WRN_QUALITY_INVALID_AREA_HORIZONTALLY

```
#define IBSU_WRN_QUALITY_INVALID_AREA_HORIZONTALLY (IBSU_WRN_QUALITY_INVALID_AREA | 1)
```

Finger was located on the horizontal invalid area

Definition at line 423 of file IBScanUltimateApi_err.h.

12.65.2.3 IBSU_WRN_QUALITY_INVALID_AREA_VERTICALLY

```
#define IBSU_WRN_QUALITY_INVALID_AREA_VERTICALLY (IBSU_WRN_QUALITY_INVALID_AREA | 2)
```

Finger was located on the vertical invalid area

Definition at line 425 of file IBScanUltimateApi_err.h.

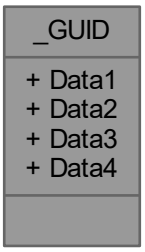
Chapter 13

Class Documentation

13.1 _GUID Struct Reference

`#include <LinuxPort.h>`

Collaboration diagram for _GUID:



Public Attributes

- DWORD Data1
- USHORT Data2
- USHORT Data3
- BYTE Data4 [8]

13.1.1 Detailed Description

Definition at line 54 of file LinuxPort.h.

13.1.2 Member Data Documentation

13.1.2.1 Data1

```
DWORD _GUID::Data1
```

Definition at line 55 of file LinuxPort.h.

13.1.2.2 Data2

```
USHORT _GUID::Data2
```

Definition at line 56 of file LinuxPort.h.

13.1.2.3 Data3

```
USHORT _GUID::Data3
```

Definition at line 57 of file LinuxPort.h.

13.1.2.4 Data4

```
BYTE _GUID::Data4[8]
```

Definition at line 58 of file LinuxPort.h.

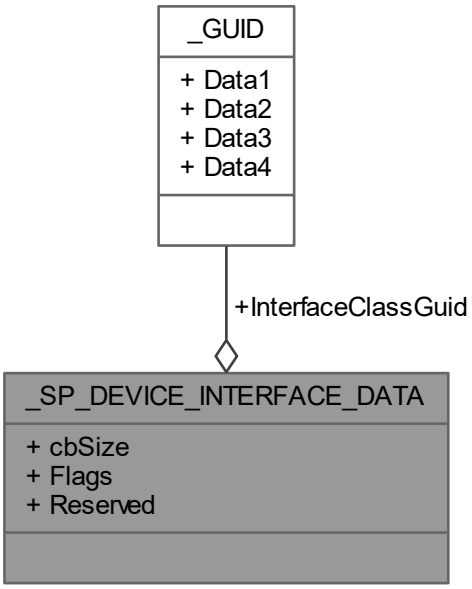
The documentation for this struct was generated from the following file:

- C_IncludeFiles/LinuxPort.h

13.2 _SP_DEVICE_INTERFACE_DATA Struct Reference

`#include <LinuxPort.h>`

Collaboration diagram for _SP_DEVICE_INTERFACE_DATA:



Public Attributes

- DWORD cbSize
- GUID InterfaceClassGuid
- DWORD Flags
- ULONG_PTR Reserved

13.2.1 Detailed Description

Definition at line 86 of file LinuxPort.h.

13.2.2 Member Data Documentation

13.2.2.1 cbSize

`DWORD _SP_DEVICE_INTERFACE_DATA::cbSize`

Definition at line 87 of file LinuxPort.h.

13.2.2.2 Flags

`DWORD _SP_DEVICE_INTERFACE_DATA::Flags`

Definition at line 89 of file LinuxPort.h.

13.2.2.3 InterfaceClassGuid

`GUID _SP_DEVICE_INTERFACE_DATA::InterfaceClassGuid`

Definition at line 88 of file LinuxPort.h.

13.2.2.4 Reserved

`ULONG_PTR _SP_DEVICE_INTERFACE_DATA::Reserved`

Definition at line 90 of file LinuxPort.h.

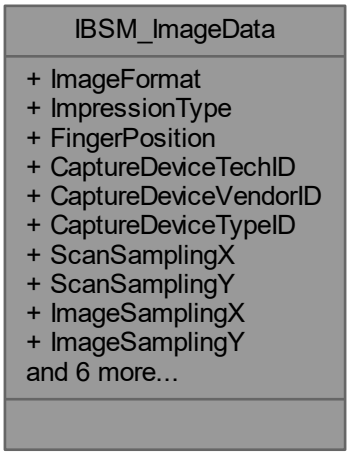
The documentation for this struct was generated from the following file:

- C_IncludeFiles/LinuxPort.h

13.3 IBSM_ImageData Struct Reference

`#include <IBScanUltimateApi_defs.h>`

Collaboration diagram for IBSM_ImageData:



Public Attributes

- 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

13.3.1 Detailed Description

Definition at line 1606 of file IBScanUltimateApi_defs.h.

13.3.2 Member Data Documentation

13.3.2.1 BitDepth

```
unsigned char IBSM_ImageData::BitDepth
```

Definition at line 1621 of file IBScanUltimateApi_defs.h.

13.3.2.2 CaptureDeviceTechID

```
IBSM_CaptureDeviceTechID IBSM_ImageData::CaptureDeviceTechID
```

Definition at line 1611 of file IBScanUltimateApi_defs.h.

13.3.2.3 CaptureDeviceTypeID

```
unsigned short IBSM_ImageData::CaptureDeviceTypeID
```

Definition at line 1613 of file IBScanUltimateApi_defs.h.

13.3.2.4 CaptureDeviceVendorID

```
unsigned short IBSM_ImageData::CaptureDeviceVendorID
```

Definition at line 1612 of file IBScanUltimateApi_defs.h.

13.3.2.5 FingerPosition

```
IBSM_FingerPosition IBSM_ImageData::FingerPosition
```

Definition at line 1610 of file IBScanUltimateApi_defs.h.

13.3.2.6 ImageData

```
void* IBSM_ImageData::ImageData
```

Definition at line 1623 of file IBScanUltimateApi_defs.h.

13.3.2.7 ImageDataLength

```
unsigned int IBSM_ImageData::ImageDataLength
```

Definition at line 1622 of file IBScanUltimateApi_defs.h.

13.3.2.8 ImageFormat

```
IBSM_ImageFormat IBSM_ImageData::ImageFormat
```

Definition at line 1608 of file IBScanUltimateApi_defs.h.

13.3.2.9 ImageSamplingX

```
unsigned short IBSM_ImageData::ImageSamplingX
```

Definition at line 1616 of file IBScanUltimateApi_defs.h.

13.3.2.10 ImageSamplingY

```
unsigned short IBSM_ImageData::ImageSamplingY
```

Definition at line 1617 of file IBScanUltimateApi_defs.h.

13.3.2.11 ImageSizeX

```
unsigned short IBSM_ImageData::ImageSizeX
```

Definition at line 1618 of file IBScanUltimateApi_defs.h.

13.3.2.12 ImageSizeY

```
unsigned short IBSM_ImageData::ImageSizeY
```

Definition at line 1619 of file IBScanUltimateApi_defs.h.

13.3.2.13 ImpressionType

```
IBSM_ImpressionType IBSM_ImageData::ImpressionType
```

Definition at line 1609 of file IBScanUltimateApi_defs.h.

13.3.2.14 ScaleUnit

```
unsigned char IBSM_ImageData::ScaleUnit
```

Definition at line 1620 of file IBScanUltimateApi_defs.h.

13.3.2.15 ScanSamplingX

```
unsigned short IBSM_ImageData::ScanSamplingX
```

Definition at line 1614 of file IBScanUltimateApi_defs.h.

13.3.2.16 ScanSamplingY

```
unsigned short IBSM_ImageData::ScanSamplingY
```

Definition at line 1615 of file IBScanUltimateApi_defs.h.

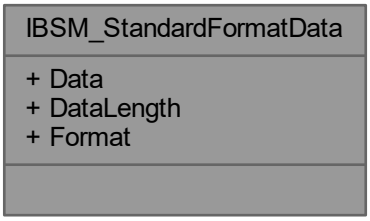
The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimateApi_defs.h

13.4 IBSM_StandardFormatData Struct Reference

```
#include <IBScanUltimateApi_defs.h>
```

Collaboration diagram for IBSM_StandardFormatData:



Public Attributes

- void * Data
- unsigned long DataLength
- IBSM_StandardFormat Format

13.4.1 Detailed Description

Definition at line 1741 of file IBScanUltimateApi_defs.h.

13.4.2 Member Data Documentation

13.4.2.1 Data

```
void* IBSM_StandardFormatData::Data
```

Pointer to data 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.

Definition at line 1746 of file IBScanUltimateApi_defs.h.

13.4.2.2 DataLength

```
unsigned long IBSM_StandardFormatData::DataLength
```

Data Length (in bytes).

Definition at line 1749 of file IBScanUltimateApi_defs.h.

13.4.2.3 Format

```
IBSM_StandardFormat IBSM_StandardFormatData::Format
```

Standard Format (ISO 19794-2:2005, ISO 19794-4:2005, ISO 19794-2:2011, ISO 19794-4:2011, ANSI/INCITS 378:2004, ANSI/INCITS 381:2004)

Definition at line 1752 of file IBScanUltimateApi_defs.h.

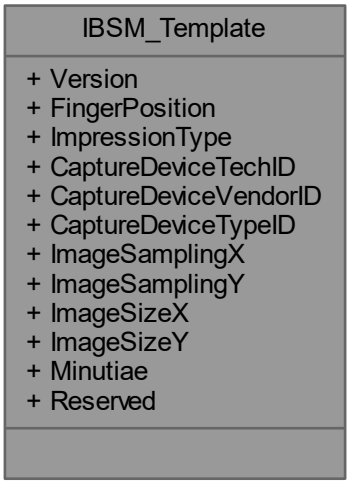
The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimateApi_defs.h

13.5 IBSM_Template Struct Reference

```
#include <IBScanUltimateApi_defs.h>
```

Collaboration diagram for IBSM_Template:



Public Attributes

- IBSM_TemplateVersion Version
- unsigned int FingerPosition
- IBSM_ImpressionType ImpressionType
- IBSM_CaptureDeviceTechID CaptureDeviceTechID
- unsigned short CaptureDeviceVendorID
- unsigned short CaptureDeviceTypeID
- unsigned short ImageSamplingX
- unsigned short ImageSamplingY
- unsigned short ImageSizeX
- unsigned short ImageSizeY
- unsigned int Minutiae [IBSU_MAX_MINUTIAE_SIZE]
- unsigned int Reserved

13.5.1 Detailed Description

Definition at line 1670 of file IBScanUltimateApi_defs.h.

13.5.2 Member Data Documentation

13.5.2.1 CaptureDeviceTechID

```
IBSM_CaptureDeviceTechID IBSM_Template::CaptureDeviceTechID
```

Definition at line 1675 of file IBScanUltimateApi_defs.h.

13.5.2.2 CaptureDeviceTypeID

```
unsigned short IBSM_Template::CaptureDeviceTypeID
```

WATSON : 0x0001, SHERLOCK : 0x0010, WATSON_MINI : 0x0020, COLUMBO : 0x0030, HOLMES : 0x0040

Definition at line 1679 of file IBScanUltimateApi_defs.h.

13.5.2.3 CaptureDeviceVendorID

```
unsigned short IBSM_Template::CaptureDeviceVendorID
```

Integrated Biometrics (0x007B)

Definition at line 1677 of file IBScanUltimateApi_defs.h.

13.5.2.4 FingerPosition

```
unsigned int IBSM_Template::FingerPosition
```

Definition at line 1673 of file IBScanUltimateApi_defs.h.

13.5.2.5 ImageSamplingX

```
unsigned short IBSM_Template::ImageSamplingX
```

500 dpi

Definition at line 1681 of file IBScanUltimateApi_defs.h.

13.5.2.6 ImageSamplingY

```
unsigned short IBSM_Template::ImageSamplingY
```

500 dpi

Definition at line 1683 of file IBScanUltimateApi_defs.h.

13.5.2.7 ImageSizeX

```
unsigned short IBSM_Template::ImageSizeX
```

Definition at line 1684 of file IBScanUltimateApi_defs.h.

13.5.2.8 ImageSizeY

```
unsigned short IBSM_Template::ImageSizeY
```

Definition at line 1685 of file IBScanUltimateApi_defs.h.

13.5.2.9 ImpressionType

```
IBSM_ImpressionType IBSM_Template::ImpressionType
```

Definition at line 1674 of file IBScanUltimateApi_defs.h.

13.5.2.10 Minutiae

```
unsigned int IBSM_Template::Minutiae[IBSU_MAX_MINUTIAE_SIZE]
```

Definition at line 1686 of file IBScanUltimateApi_defs.h.

13.5.2.11 Reserved

```
unsigned int IBSM_Template::Reserved
```

Definition at line 1687 of file IBScanUltimateApi_defs.h.

13.5.2.12 Version

```
IBSM_TemplateVersion IBSM_Template::Version
```

Definition at line 1672 of file IBScanUltimateApi_defs.h.

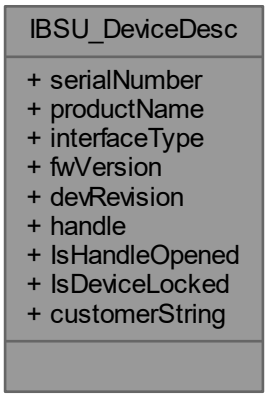
The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimateApi_defs.h

13.6 IBSU_DeviceDesc Struct Reference

```
#include <IBScanUltimateApi_defs.h>
```

Collaboration diagram for IBSU_DeviceDesc:



Public Attributes

- char serialNumber [IBSU_MAX_STR_LEN]
- char productName [IBSU_MAX_STR_LEN]
- char interfaceType [IBSU_MAX_STR_LEN]
- char fwVersion [IBSU_MAX_STR_LEN]
- char devRevision [IBSU_MAX_STR_LEN]
- int handle
- BOOL IsHandleOpened
- BOOL IsDeviceLocked
- char customerString [IBSU_MAX_STR_LEN]

13.6.1 Detailed Description

Definition at line 436 of file IBScanUltimateApi_defs.h.

13.6.2 Member Data Documentation

13.6.2.1 customerString

```
char IBSU_DeviceDesc::customerString[IBSU_MAX_STR_LEN]
```

Definition at line 459 of file IBScanUltimateApi_defs.h.

13.6.2.2 devRevision

```
char IBSU_DeviceDesc::devRevision[IBSU_MAX_STR_LEN]
```

Device revision.

Definition at line 447 of file IBScanUltimateApi_defs.h.

13.6.2.3 fwVersion

```
char IBSU_DeviceDesc::fwVersion[IBSU_MAX_STR_LEN]
```

Device firmware version.

Definition at line 445 of file IBScanUltimateApi_defs.h.

13.6.2.4 handle

```
int IBSU_DeviceDesc::handle
```

Return device handle.

Definition at line 449 of file IBScanUltimateApi_defs.h.

13.6.2.5 interfaceType

```
char IBSU_DeviceDesc::interfaceType[IBSU_MAX_STR_LEN]
```

Device interface type (USB, Firewire).

Definition at line 443 of file IBScanUltimateApi_defs.h.

13.6.2.6 IsDeviceLocked

```
BOOL IBSU_DeviceDesc::IsDeviceLocked
```

Definition at line 457 of file IBScanUltimateApi_defs.h.

13.6.2.7 IsHandleOpened

```
BOOL IBSU_DeviceDesc::IsHandleOpened
```

Check if device handle is opened.

Definition at line 451 of file IBScanUltimateApi_defs.h.

13.6.2.8 productName

```
char IBSU_DeviceDesc::productName[IBSU_MAX_STR_LEN]
```

Device product name.

Definition at line 441 of file IBScanUltimateApi_defs.h.

13.6.2.9 serialNumber

```
char IBSU_DeviceDesc::serialNumber[IBSU_MAX_STR_LEN]
```

Device serial number.

Definition at line 439 of file IBScanUltimateApi_defs.h.

The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimateApi_defs.h

13.7 IBSU_ImageData Struct Reference

```
#include <IBScanUltimate.h>
```

Collaboration diagram for IBSU_ImageData:

Public Attributes

- void * Buffer
- DWORD Width
- DWORD Height
- double ResolutionX
- double ResolutionY
- double FrameTime
- int Pitch
- BYTE BitsPerPixel
- IBSU_ImageFormat Format
- BOOL IsFinal
- DWORD ProcessThres

13.7.1 Detailed Description

Definition at line 73 of file IBScanUltimate.h.

13.7.2 Member Data Documentation

13.7.2.1 BitsPerPixel

```
BYTE IBSU_ImageData::BitsPerPixel
```

Definition at line 100 of file IBScanUltimate.h.

13.7.2.2 Buffer

```
void* IBSU_ImageData::Buffer
```

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.

Definition at line 78 of file IBScanUltimate.h.

13.7.2.3 Format

```
IBSU_ImageFormat IBSU_ImageData::Format
```

Image color format.

Definition at line 103 of file IBScanUltimate.h.

13.7.2.4 FrameTime

```
double IBSU_ImageData::FrameTime
```

Image acquisition time, excluding processing time (in seconds).

Definition at line 93 of file IBScanUltimate.h.

13.7.2.5 Height

```
DWORD IBSU_ImageData::Height
```

Image vertical size (in pixels).

Definition at line 84 of file IBScanUltimate.h.

13.7.2.6 IsFinal

```
BOOL IBSU_ImageData::IsFinal
```

Marks image as the final processed result from the capture. If this is FALSE, the image is a preview image or a preliminary result.

Definition at line 107 of file IBScanUltimate.h.

13.7.2.7 Pitch

```
int IBSU_ImageData::Pitch
```

Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order.

Definition at line 97 of file IBScanUltimate.h.

13.7.2.8 ProcessThres

```
DWORD IBSU_ImageData::ProcessThres
```

Threshold of image processing.

Definition at line 110 of file IBScanUltimate.h.

13.7.2.9 ResolutionX

```
double IBSU_ImageData::ResolutionX
```

Horizontal image resolution (in pixels/inch).

Definition at line 87 of file IBScanUltimate.h.

13.7.2.10 ResolutionY

```
double IBSU_ImageData::ResolutionY
```

Vertical image resolution (in pixels/inch).

Definition at line 90 of file IBScanUltimate.h.

13.7.2.11 Width

```
DWORD IBSU_ImageData::Width
```

Image horizontal size (in pixels).

Definition at line 81 of file IBScanUltimate.h.

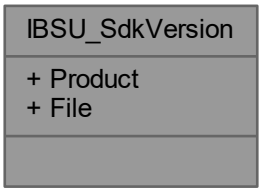
The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimate.h

13.8 IBSU_SdkVersion Struct Reference

```
#include <IBScanUltimateApi_defs.h>
```

Collaboration diagram for IBSU_SdkVersion:



Public Attributes

- char Product [IBSU_MAX_STR_LEN]
- char File [IBSU_MAX_STR_LEN]

13.8.1 Detailed Description

Definition at line 399 of file IBScanUltimateApi_defs.h.

13.8.2 Member Data Documentation

13.8.2.1 File

```
char IBSU_SdkVersion::File[IBSU_MAX_STR_LEN]
```

File version string.

Definition at line 404 of file IBScanUltimateApi_defs.h.

13.8.2.2 Product

```
char IBSU_SdkVersion::Product[IBSU_MAX_STR_LEN]
```

Product version string.

Definition at line 402 of file IBScanUltimateApi_defs.h.

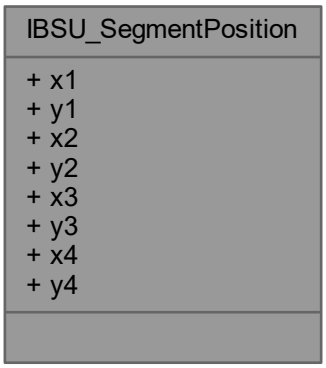
The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimateApi_defs.h

13.9 IBSU_SegmentPosition Struct Reference

```
#include <IBScanUltimateApi_defs.h>
```

Collaboration diagram for IBSU_SegmentPosition:



Public Attributes

- short x1
- short y1
- short x2
- short y2
- short x3
- short y3
- short x4
- short y4

13.9.1 Detailed Description

Definition at line 546 of file IBScanUltimateApi_defs.h.

13.9.2 Member Data Documentation

13.9.2.1 x1

```
short IBSU_SegmentPosition::x1
```

X coordinate of starting point of the finger segment.

Definition at line 549 of file IBScanUltimateApi_defs.h.

13.9.2.2 x2

```
short IBSU_SegmentPosition::x2
```

X coordinate of 1st corner of the finger segment.

Definition at line 553 of file IBScanUltimateApi_defs.h.

13.9.2.3 x3

```
short IBSU_SegmentPosition::x3
```

X coordinate of 2nd corner of the finger segment.

Definition at line 557 of file IBScanUltimateApi_defs.h.

13.9.2.4 x4

```
short IBSU_SegmentPosition::x4
```

X coordinate of 3rd corner of the finger segment.

Definition at line 561 of file IBScanUltimateApi_defs.h.

13.9.2.5 y1

```
short IBSU_SegmentPosition::y1
```

Y coordinate of starting point of the finger segment.

Definition at line 551 of file IBScanUltimateApi_defs.h.

13.9.2.6 y2

```
short IBSU_SegmentPosition::y2
```

Y coordinate of 1st corner of the finger segment.

Definition at line 555 of file IBScanUltimateApi_defs.h.

13.9.2.7 y3

```
short IBSU_SegmentPosition::y3
```

Y coordinate of 2nd corner of the finger segment.

Definition at line 559 of file IBScanUltimateApi_defs.h.

13.9.2.8 y4

```
short IBSU_SegmentPosition::y4
```

Y coordinate of 3rd corner of the finger segment.

Definition at line 563 of file IBScanUltimateApi_defs.h.

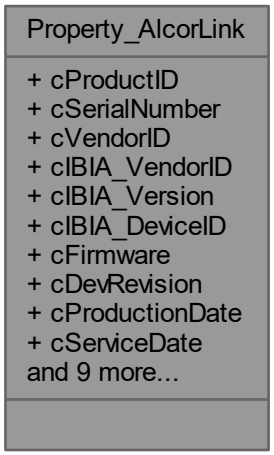
The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimateApi_defs.h

13.10 Property_AlcorLink Struct Reference

`#include <IBScanUltimateApi_defs.h>`

Collaboration diagram for Property_AlcorLink:



Public Attributes

- char cProductID [PARALLEL_MAX_STR_LEN]
- char cSerialNumber [PARALLEL_MAX_STR_LEN]
- char cVendorID [PARALLEL_MAX_STR_LEN]
- char cIBIA_VendorID [PARALLEL_MAX_STR_LEN]
- char cIBIA_Version [PARALLEL_MAX_STR_LEN]
- char cIBIA_DeviceID [PARALLEL_MAX_STR_LEN]
- char cFirmware [PARALLEL_MAX_STR_LEN]
- char cDevRevision [PARALLEL_MAX_STR_LEN]
- char cProductionDate [PARALLEL_MAX_STR_LEN]
- char cServiceDate [PARALLEL_MAX_STR_LEN]
- char cFPGA [PARALLEL_MAX_STR_LEN]
- char cCMT1 [PARALLEL_MAX_STR_LEN]
- char cCMT2 [PARALLEL_MAX_STR_LEN]
- char cCMT3 [PARALLEL_MAX_STR_LEN]
- char cCMT4 [PARALLEL_MAX_STR_LEN]
- unsigned short idVendor
- unsigned short idProduct
- char cCalibrationData_str1 [PARALLEL_MAX_STR_LEN]
- char cCalibrationData_str2 [PARALLEL_MAX_STR_LEN]

13.10.1 Detailed Description

Definition at line 509 of file IBScanUltimateApi_defs.h.

13.10.2 Member Data Documentation

13.10.2.1 cCalibrationData_str1

```
char Property_AlcorLink::cCalibrationData_str1[PARALLEL_MAX_STR_LEN]
```

Definition at line 528 of file IBScanUltimateApi_defs.h.

13.10.2.2 cCalibrationData_str2

```
char Property_AlcorLink::cCalibrationData_str2[PARALLEL_MAX_STR_LEN]
```

Definition at line 529 of file IBScanUltimateApi_defs.h.

13.10.2.3 cCMT1

```
char Property_AlcorLink::cCMT1[PARALLEL_MAX_STR_LEN]
```

Definition at line 522 of file IBScanUltimateApi_defs.h.

13.10.2.4 cCMT2

```
char Property_AlcorLink::cCMT2[PARALLEL_MAX_STR_LEN]
```

Definition at line 523 of file IBScanUltimateApi_defs.h.

13.10.2.5 cCMT3

```
char Property_AlcorLink::cCMT3[PARALLEL_MAX_STR_LEN]
```

Definition at line 524 of file IBScanUltimateApi_defs.h.

13.10.2.6 cCMT4

```
char Property_AlcorLink::cCMT4[PARALLEL_MAX_STR_LEN]
```

Definition at line 525 of file IBScanUltimateApi_defs.h.

13.10.2.7 cDevRevision

```
char Property_AlcorLink::cDevRevision[PARALLEL_MAX_STR_LEN]
```

Definition at line 518 of file IBScanUltimateApi_defs.h.

13.10.2.8 cFirmware

```
char Property_AlcorLink::cFirmware[PARALLEL_MAX_STR_LEN]
```

Definition at line 517 of file IBScanUltimateApi_defs.h.

13.10.2.9 cFPGA

```
char Property_AlcorLink::cFPGA[PARALLEL_MAX_STR_LEN]
```

Definition at line 521 of file IBScanUltimateApi_defs.h.

13.10.2.10 cIBIA_DeviceID

```
char Property_AlcorLink::cIBIA_DeviceID[PARALLEL_MAX_STR_LEN]
```

Definition at line 516 of file IBScanUltimateApi_defs.h.

13.10.2.11 cIBIA_VendorID

```
char Property_AlcorLink::cIBIA_VendorID[PARALLEL_MAX_STR_LEN]
```

Definition at line 514 of file IBScanUltimateApi_defs.h.

13.10.2.12 cIBIA_Version

```
char Property_AlcorLink::cIBIA_Version[PARALLEL_MAX_STR_LEN]
```

Definition at line 515 of file IBScanUltimateApi_defs.h.

13.10.2.13 cProductID

```
char Property_AlcorLink::cProductID[PARALLEL_MAX_STR_LEN]
```

Definition at line 511 of file IBScanUltimateApi_defs.h.

13.10.2.14 cProductionDate

```
char Property_AlcorLink::cProductionDate[PARALLEL_MAX_STR_LEN]
```

Definition at line 519 of file IBScanUltimateApi_defs.h.

13.10.2.15 cSerialNumber

```
char Property_AlcorLink::cSerialNumber[PARALLEL_MAX_STR_LEN]
```

Definition at line 512 of file IBScanUltimateApi_defs.h.

13.10.2.16 cServiceDate

```
char Property_AlcorLink::cServiceDate[PARALLEL_MAX_STR_LEN]
```

Definition at line 520 of file IBScanUltimateApi_defs.h.

13.10.2.17 cVendorID

```
char Property_AlcorLink::cVendorID[PARALLEL_MAX_STR_LEN]
```

Definition at line 513 of file IBScanUltimateApi_defs.h.

13.10.2.18 idProduct

```
unsigned short Property_AlcorLink::idProduct
```

Definition at line 527 of file IBScanUltimateApi_defs.h.

13.10.2.19 idVendor

```
unsigned short Property_AlcorLink::idVendor
```

Definition at line 526 of file IBScanUltimateApi_defs.h.

The documentation for this struct was generated from the following file:

- C_IncludeFiles/IBScanUltimateApi_defs.h

Chapter 14

File Documentation

14.1 C_IncludeFiles/ApiFunctions.dox File Reference

14.2 C_IncludeFiles/HowTo - Duplicate_Finger.dox File Reference

14.3 C_IncludeFiles/HowTo - Encryption.dox File Reference

14.4 C_IncludeFiles/HowTo - Hand_Checker.dox File Reference

14.5 C_IncludeFiles/HowTo - NFIQ2.dox File Reference

14.6 C_IncludeFiles/HowTo - Required_SDK.dox File Reference

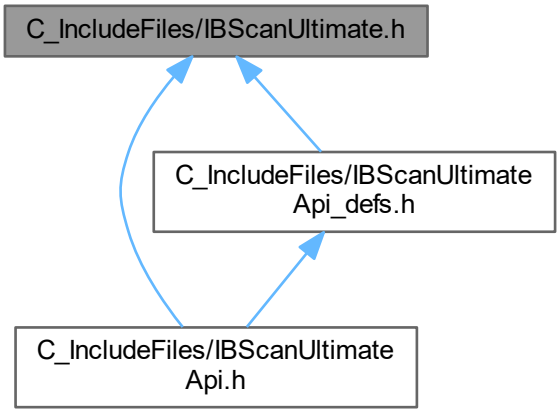
14.7 C_IncludeFiles/HowTo - Spoof Function.dox File Reference

14.8 C_IncludeFiles/HowTo.dox File Reference

14.9 C_IncludeFiles/IBScanUltimate.h File Reference

Definition of image data structures for IBScanUltimate. https://integratedbiometrics.com/.

This graph shows which files directly or indirectly include this file:



Classes

- struct IBSU_ImageData

Enumerations

- enum IBSU_ImageFormat { IBSU_IMG_FORMAT_GRAY , IBSU_IMG_FORMAT_RGB24 , IBSU_IMG_FORMAT_RGB32 , IBSU_IMG_FORMAT_UNKNOWN }

14.9.1 Detailed Description

Definition of image data structures for IBScanUltimate. https://integratedbiometrics.com/.

IBScanUltimate.h

Author

Integrated Biometrics, LLC

Copyright

Copyright (c) Integrated Biometrics, 2009-2022
http://www.integratedbiometrics.com

Definition in file IBScanUltimate.h.

14.10 IBScanUltimate.h

Go to the documentation of this file.

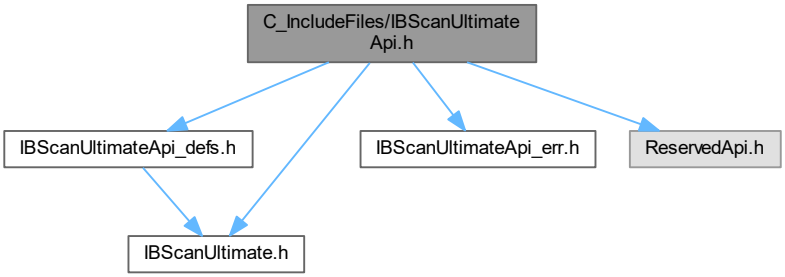
```
00001
00028 #pragma once
00029
00030 #ifdef __linux__
00031 #include "LinuxPort.h"
00032 #endif
00033
00034 #ifdef __cplusplus
00035 extern "C" {
00036 #endif
00037
00038
00046     typedef enum
00047     {
00049         IBSU_IMG_FORMAT_GRAY,
00051         IBSU_IMG_FORMAT_RGB24,
00053         IBSU_IMG_FORMAT_RGB32,
00055         IBSU_IMG_FORMAT_UNKNOWN
00056     }
00057     IBSU_ImageFormat;
00073     typedef struct
00074     {
00078         void                *Buffer;
00079
00081         DWORD               Width;
00082
00084         DWORD               Height;
00085
00087         double              ResolutionX;
00088
00090         double              ResolutionY;
00091
00093         double              FrameTime;
00094
00097         int                 Pitch;
00098
00099         /* Number of bits per pixel. */
00100         BYTE                BitsPerPixel;
00101
00103         IBSU_ImageFormat    Format;
00104
00107         BOOL                IsFinal;
00108
00110         DWORD               ProcessThres;
00111     }IBSU_ImageData;
00118 #ifdef __cplusplus
00119 } // extern "C"
00120 #endif
```

14.11 C_IncludeFiles/IBScanUltimateApi.h File Reference

API functions for IBScanUltimate.

```
#include "IBScanUltimateApi_defs.h"
#include "IBScanUltimateApi_err.h"
#include "IBScanUltimate.h"
#include "ReservedApi.h"
```

Include dependency graph for IBScanUltimateApi.h:



Functions

- int WINAPI IBSU_OpenDevice (const int deviceIndex, int *pHandle)

  Initializes a device, given a particular device index.
- int WINAPI IBSU_OpenDeviceEx (const int deviceIndex, LPCSTR uniformityMaskPath, const BOOL asyncOpen, int *pHandle)

  Extension of initialize device(fast mode), given by a particular device index.
- int WINAPI IBSU_AsyncOpenDevice (const int deviceIndex)

  Asynchronous Initialize device, given by a particular device index.
- int WINAPI IBSU_CloseDevice (const int handle)

  Releases a device (by device handle).
- int WINAPI IBSU_CloseAllDevice ()

  Releases all currently initialized devices (particular device handle not needed).
- int WINAPI IBSU_IsDeviceOpened (const int handle)

  Check if a particular device is opened/initialized.
- int WINAPI IBSU_GetDeviceCount (int *pDeviceCount)

  Retrieve the number of connected IB USB devices. Device count function trigger for Android SDK (Reduce polling count)
- int WINAPI IBSU_GetDeviceDescription (const int deviceIndex, IBSU_DeviceDesc *pDeviceDesc)

  Retrieve detailed device information about a particular scanner by its logical index.
- int WINAPI IBSU_GetRequiredSDKVersion (const int deviceIndex, LPSTR minSDKVersion)

  Get minimum SDK version required for running.
- int WINAPI IBSU_SetProperty (const int handle, const IBSU_PropertyId propertyId, LPCSTR propertyValue)

  Set a device's property value (by handle).
- int WINAPI IBSU_GetProperty (const int handle, const IBSU_PropertyId propertyId, LPSTR propertyValue)

  Retrieves a particular device's property value (by handle).
- int WINAPI IBSU_IsCaptureAvailable (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, BOOL *pIsAvailable)

  Check if a requested capture mode is supported by the device.
- int WINAPI IBSU_BeginCaptureImage (const int handle, const IBSU_ImageType imageType, const IBSU_ImageResolution imageResolution, const DWORD captureOptions)

  Starts image acquisition for a particular device (by handle).
- int WINAPI IBSU_CancelCaptureImage (const int handle)

  Abort image acquisition on a device that is currently scanning.
- int WINAPI IBSU_IsCaptureActive (const int handle, BOOL *pIsActive)

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

- int WINAPI IBSU_TakeResultImageManually (const int handle)

  Start image acquisition for a particular device (by handle) with image gain manually set.
- int WINAPI IBSU_GetIBSM_ResultImageInfo (const int handle, IBSM_FingerPosition fingerPosition, IBSM_ImageData *pResultImage, IBSM_ImageData *pSplitResultImage, int *pSplitResultImageCount)

  Get the result image information.
- int WINAPI IBSU_IsTouchedFinger (const int handle, int *pTouchInValue)

  Queries a particular scanner to determine if a finger is currently detected.
- int WINAPI IBSU_CheckWetFinger (const int handle, const IBSU_ImageData inImage)

  Check if the image is wet or not.
- int WINAPI IBSU_GetImageWidth (const int handle, const IBSU_ImageData inImage, int *Width_MM)

  Get the image width of input image by millimeter(mm).
- int WINAPI IBSU_ConvertImageToISOANSI (const int handle, const IBSM_ImageData *image, const int imageCount, const IBSM_ImageFormat imageFormat, const IBSM_StandardFormat STDformat, IBSM_StandardFormatData *pdata)

  Convert Image Data to Standard Format for write file. (ISO 19794-2:2005, ISO 19794-4:2005, ISO 19794-2:2011, ISO 19794-4:2011, ANSI/INCITS 378:2004, ANSI/INCITS 381:2004)
- int WINAPI IBSU_GetContrast (const int handle, int *pContrastValue)

  Get the contrast value for a particular scanner.
- int WINAPI IBSU_SetContrast (const int handle, const int contrastValue)

  Set the contrast value for a particular scanner.
- int WINAPI IBSU_SetLEOperationMode (const int handle, const IBSU_LEOperationMode leOperationMode)

  Sets the LE operation mode (On, Off, or Auto) for a particular scanner.
- int WINAPI IBSU_GetLEOperationMode (const int handle, IBSU_LEOperationMode *pLeOperationMode)

  Get the light-emitting (LE) film operation mode for a device.
- int WINAPI IBSU_GetOperableLEDs (const int handle, IBSU_LedType *pLedType, int *pLedCount, DWORD *pOperableLEDs)

  Get operable status LED's.
- int WINAPI IBSU_GetLEDs (const int handle, DWORD *pActiveLEDs)

  Get active status LED's for a particular scanner.
- int WINAPI IBSU_SetLEDs (const int handle, const DWORD activeLEDs)

  Set active status LED's on a particular scanner.
- int WINAPI IBSU_GetOperableBeeper (const int handle, IBSU_BeeperType *pBeeperType)
- 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)

  Set the value of Beeper on a device.
- int WINAPI IBSU_BGetImage (const int handle, IBSU_ImageData *pImage, IBSU_ImageType *pImageType, IBSU_ImageData *pSplitImageArray, int *pSplitImageArrayCount, IBSU_FingerCountState *pFingerCount↩State, IBSU_FingerQualityState *pQualityArray, int *pQualityArrayCount)

  Acquire an image from a device, blocking for result. The split 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.
- 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)

  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.
- int WINAPI IBSU_BGetInitProgress (const int deviceIndex, BOOL *pIsComplete, int *pHandle, int *p↩ProgressValue)

  Get initialization progress of a device. If initialization is complete, the handle for subsequent function calls will be returned to the application.
- int WINAPI IBSU_BGetClearPlatenAtCapture (const int handle, IBSU_PlatenState *pPlatenState)

  Determine whether the platen was clear when capture was started or has since become clear.

- int WINAPI IBSU_BGetRollingInfo (const int handle, IBSU_RollingState *pRollingState, int *pRollingLineX)

  Get information about the status of the rolled print capture for a device.
- int WINAPI IBSU_BGetRollingInfoEx (const int handle, IBSU_RollingState *pRollingState, int *pRollingLineX, int *pRollingDirection, int *pRollingWidth)

  Get information about the status of the rolled print capture for a device.
- int WINAPI IBSU_GenerateZoomOutImage (const IBSU_ImageData inImage, BYTE *outImage, const int outWidth, const int outHeight, const BYTE bkColor)

  Generate scaled version of an image.
- int WINAPI IBSU_GenerateZoomOutImageEx (const BYTE *pInImage, const int inWidth, const int inHeight, BYTE *outImage, const int outWidth, const int outHeight, const BYTE bkColor)

  Generate scaled version of an image.
- int WINAPI IBSU_SaveBitmapImage (LPCSTR filePath, const BYTE *imgBuffer, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)

  Save image to bitmap file.
- 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)

  Save image to bitmap memory.
- int WINAPI IBSU_WSQEncodeMem (const BYTE *image, const int width, const int height, const int pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText, BYTE **compressedData, int *compressedLength)

  WSQ compresses a grayscale fingerprint image.
- int WINAPI IBSU_WSQEncodeToFile (LPCSTR filePath, const BYTE *image, const int width, const int height, const int pitch, const int bitsPerPixel, const int pixelPerInch, const double bitRate, const char *commentText)

  Save WSQ compressed grayscale fingerprint image to a specific file path.
- int WINAPI IBSU_WSQDecodeMem (const BYTE *compressedImage, const int compressedLength, BYTE **decompressedImage, int *outWidth, int *outHeight, int *outPitch, int *outBitsPerPixel, int *outPixelPerInch)

  Decompress a WSQ-encoded grayscale fingerprint image.
- int WINAPI IBSU_WSQDecodeFromFile (LPCSTR filePath, BYTE **decompressedImage, int *outWidth, int *outHeight, int *outPitch, int *outBitsPerPixel, int *outPixelPerInch)

  Decompress a WSQ-encoded grayscale fingerprint image from a specific file path.
- int WINAPI IBSU_FreeMemory (void *memblock)

  Release the allocated memory block from the internal heap of the library. This is obtained by IBSU_WSQEncodeMem(), IBSU_WSQDecodeMem, IBSU_WSQDecodeFromFile() and other API functions.
- int WINAPI IBSU_SavePngImage (LPCSTR filePath, const BYTE *image, const DWORD width, const DWORD height, const int pitch, const double resX, const double resY)

  Save image to PNG file.
- 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)

  Save image to JPEG-2000 file.
- int WINAPI IBSU_CombineImage (const IBSU_ImageData inImage1, const IBSU_ImageData inImage2, IBSU_CombineImageWhichHand whichHand, IBSU_ImageData *outImage)

  Combine two images (2 flat fingers) into a single image (left/right hands)
- int WINAPI IBSU_CombineImageEx (const IBSU_ImageData InImage1, const IBSU_ImageData InImage2, IBSU_CombineImageWhichHand WhichHand, IBSU_ImageData *OutImage, IBSU_ImageData *pSegmentImageArray, IBSU_SegmentPosition *pSegmentPositionArray, int *pSegmentImageArrayCount)

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

  Generate scaled image in various formats for fast image display on canvas. This can be used instead of IBSU_GenerateZoomOutImageEx()

- int WINAPI IBSU_RemoveFingerImage (const int handle, const DWORD fIndex)
- int WINAPI IBSU_AddFingerImage (const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const BOOL flagForce)

  Add a finger image for the fingerprint duplicate check and roll to slap comparison. It can have only ten prints.
- int WINAPI IBSU_IsFingerDuplicated (const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, const int securityLevel, DWORD *pMatchedPosition)

  Checks for a fingerprint duplicate from the stored prints by IBSU_AddFingerImage().
- int WINAPI IBSU_IsValidFingerGeometry (const int handle, const IBSU_ImageData image, const DWORD fIndex, const IBSU_ImageType imageType, BOOL *pValid)

  Check for hand and finger geometry whether it is correct or not.
- int WINAPI IBSU_GetNFIQScore (const int handle, const BYTE *imgBuffer, const DWORD width, const DWORD height, const BYTE bitsPerPixel, int *pScore)

  Calculate NFIQ score for an image.
- int WINAPI IBSU_GetNFIQScoreEx (const int handle, const BYTE *imgBuffer, const DWORD width, const DWORD height, const int pitch, const BYTE bitsPerPixel, int *pScore)
- int WINAPI IBSU_IsSpoofFingerDetected (const int handle, const IBSU_ImageData image, BOOL *pIsSpoof)

  Detect if the finger print is Live or Fake.
- int WINAPI IBSU_SetEncryptionKey (const int handle, const unsigned char *pEncyptionKey, const IBSU_EncryptionMode encMode)

  Set encryption key and mode. (Currently not supported)
- int WINAPI IBSU_SetCustomerKey (const int deviceIndex, const IBSU_HashType hashType, LPCSTR p↩ CustomerKey)

  Set CustomerKey to use locked devices, This must be perfomed on locked devices before IBSU_OpenDevice.
- int WINAPI IBSU_GetSDKVersion (IBSU_SdkVersion *pVerinfo)

  Gets a structure holding product and software version information (IBSU_SdkVersion).
- int WINAPI IBSU_GetSDKVersionW (IBSU_SdkVersionW *pVerinfo)
- int WINAPI IBSU_UnloadLibrary ()

  The library is unmapped from the address space explicitly, and the library is no longer valid.
- int WINAPI IBSU_IsWritableDirectory (LPCSTR dirpath, const BOOL needCreateSubFolder)

  Check whether a directory is writable.
- int WINAPI IBSU_GetErrorString (const int errorCode, LPSTR errorString)

  Returns a string description of the error code.
- int WINAPI IBSU_EnableTraceLog (const BOOL on)

  Enable or disable trace log. The trace log is enabled by default on both Windows and Android, and disabled by default on Linux.
- int WINAPI IBSU_CreateClientWindow (const int handle, const IBSU_HWND hWindow, const DWORD left, const DWORD top, const DWORD right, const DWORD bottom)

  Create a client window associated with a device. (Available only on Windows.)
- int WINAPI IBSU_DestroyClientWindow (const int handle, const BOOL clearExistingInfo)

  Destroy client window associated with a device. (Available only on Windows.)
- int WINAPI IBSU_GetClientWindowProperty (const int handle, const IBSU_ClientWindowPropertyId propertyId, LPSTR propertyValue)

  Get the value of a property for the client window associated with a device. For descriptions of properties and values, see definition of 'IBSU_ClientWindowPropertyId'. (Available only on Windows.)
- int WINAPI IBSU_GetClientWindowPropertyW (const int handle, const IBSU_ClientWindowPropertyId propertyId, wchar_t *propertyValue)
- int WINAPI IBSU_SetClientDisplayProperty (const int handle, const IBSU_ClientWindowPropertyId propertyId, LPCSTR propertyValue)

  Set the value of a property for the client window associated with a device. For descriptions of properties and values, see definition of 'IBSU_ClientWindowPropertyId'. (Available only on Windows.)
- int WINAPI IBSU_SetClientDisplayPropertyW (const int handle, const IBSU_ClientWindowPropertyId propertyId, const wchar_t *propertyValue)
- int WINAPI IBSU_SetClientWindowOverlayText (const int handle, const char *fontName, const int fontSize, const BOOL fontBold, const char *text, const int posX, const int posY, const DWORD textColor)

Set the overlay text for the client window associated with a device. (Available only on Windows.) (Deprecated)

- int WINAPI IBSU_SetClientWindowOverlayTextW (const int handle, const wchar_t *fontName, const int fontSize, const BOOL fontBold, const wchar_t *text, const int posX, const int posY, const DWORD textColor)
- int WINAPI IBSU_ShowOverlayObject (const int handle, const int overlayHandle, const BOOL show)

  Show or hide an overlay object.
- int WINAPI IBSU_ShowAllOverlayObject (const int handle, const BOOL show)

  Show or hide all overlay objects.
- int WINAPI IBSU_RemoveOverlayObject (const int handle, const int overlayHandle)

  Remove an overlay object.
- int WINAPI IBSU_RemoveAllOverlayObject (const int handle)

  Remove all overlay objects.
- 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)

  Add an overlay text for display on the window.
- int WINAPI IBSU_AddOverlayTextW (const int handle, int *pOverlayHandle, const wchar_t *fontName, const int fontSize, const BOOL fontBold, const wchar_t *text, const int posX, const int posY, const DWORD textColor)
- 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)

  Modify an existing overlay text for display on the window.
- int WINAPI IBSU_ModifyOverlayTextW (const int handle, const int overlayHandle, const wchar_t *fontName, const int fontSize, const BOOL fontBold, const wchar_t *text, const int posX, const int posY, const DWORD textColor)
- 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)

  Add an overlay line for display on the window.
- 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)

  Modify an existing line for display on the window.
- 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)

  Add an overlay quadrangle for display on the window.
- 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)

  Modify an existing quadrangle for display on the window.
- 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)

  Add an overlay shape for display on the window.
- 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)

  Modify an overlay shape for display on the window.
- int WINAPI IBSU_RedrawClientWindow (const int handle)

  Update the specified client window which is defined by IBSU_CreateClientWindow(). (Available only on Windows.)
- int WINAPI IBSU_RegisterCallbacks (const int handle, const IBSU_Events event, void *pCallbackFunction, void *pContext)

  This function is used to register callback methods, utilizing event-driven programming when the state of the scanner changes.
- int WINAPI IBSU_ReleaseCallbacks (const int handle, const IBSU_Events events)

  Unregister a callback function for a particular event.

14.11.1 Detailed Description

API functions for IBScanUltimate.

IBScanUltimateApi.h

Author

Integrated Biometrics, LLC

Copyright

Definition in file IBScanUltimateApi.h.

14.12 IBScanUltimateApi.h

Go to the documentation of this file.

```
00001
00114 #pragma once
00115
00116 #include "IBScanUltimateApi_defs.h"
00117 #include "IBScanUltimateApi_err.h"
00118 #include "IBScanUltimate.h"
00119 #include "ReservedApi.h"
00120
00121 #ifdef __cplusplus
00122 extern "C" {
00123 #endif
00124
00151     int WINAPI IBSU_OpenDevice
00152         (const int  deviceIndex,
00153         int* pHandle);
00154
00175 #ifndef WINCE
00176     int WINAPI IBSU_OpenDeviceEx
00177         (const int  deviceIndex,
00178         LPCSTR      uniformityMaskPath,
00179         const BOOL  asyncOpen,
00180         int* pHandle);
00181
00182 #else  // UNICODE
00183     int WINAPI IBSU_OpenDeviceExW
00184         (const int      deviceIndex,
00185         const wchar_t* uniformityMaskPath,
00186         const BOOL      asyncOpen,
00187         int* pHandle);
00188
00189 #define IBSU_OpenDeviceEx IBSU_OpenDeviceExW
00190 #endif
00191
00207     int WINAPI IBSU_AsyncOpenDevice
00208         (const int  deviceIndex);
00209
00227     int WINAPI IBSU_CloseDevice
00228         (const int  handle);
00229
00245     int WINAPI IBSU_CloseAllDevice();
00246
00262     int WINAPI IBSU_IsDeviceOpened
00263         (const int  handle);
00264
00293     int WINAPI IBSU_GetDeviceCount
00294         (int* pDeviceCount);
00295
00296
00310 #ifndef WINCE
00311     int WINAPI IBSU_GetDeviceDescription
```

```
00312         (const int          deviceIndex,
00313         IBSU_DeviceDesc* pDeviceDesc);
00314
00315 #else  // UNICODE
00316     int WINAPI IBSU_GetDeviceDescriptionW
00317         (const int          deviceIndex,
00318         IBSU_DeviceDescW* pDeviceDesc);
00319
00320 #define IBSU_GetDeviceDescription IBSU_GetDeviceDescriptionW
00321 #endif
00322
00338     int WINAPI IBSU_GetRequiredSDKVersion
00339         (const int  deviceIndex,
00340         LPSTR       minSDKVersion);
00341
00374 #ifndef WINCE
00375     int WINAPI IBSU_SetProperty
00376         (const int              handle,
00377         const IBSU_PropertyId   propertyId,
00378         LPCSTR                  propertyValue);
00379
00380 #else  // UNICODE
00381     int WINAPI IBSU_SetPropertyW
00382         (const int              handle,
00383         const IBSU_PropertyId   propertyId,
00384         const wchar_t* propertyValue);
00385
00386 #define IBSU_SetProperty IBSU_SetPropertyW
00387 #endif
00388
00406 #ifndef WINCE
00407     int WINAPI IBSU_GetProperty
00408         (const int              handle,
00409         const IBSU_PropertyId   propertyId,
00410         LPSTR                   propertyValue);
00411
00412 #else  // UNICODE
00413     int WINAPI IBSU_GetPropertyW
00414         (const int              handle,
00415         const IBSU_PropertyId   propertyId,
00416         wchar_t* propertyValue);
00417
00418 #define IBSU_GetProperty IBSU_GetPropertyW
00419 #endif
00420
00452     int WINAPI IBSU_IsCaptureAvailable
00453         (const int                      handle,
00454         const IBSU_ImageType        imageType,
00455         const IBSU_ImageResolution  imageResolution,
00456         BOOL                        *pIsAvailable);
00457
00482     int WINAPI IBSU_BeginCaptureImage
00483         (const int                      handle,
00484         const IBSU_ImageType        imageType,
00485         const IBSU_ImageResolution  imageResolution,
00486         const DWORD                 captureOptions);
00487
00504     int WINAPI IBSU_CancelCaptureImage
00505         (const int  handle);
00506
00523     int WINAPI IBSU_IsCaptureActive
00524         (const int  handle,
00525         BOOL        *pIsActive);
00526
00542     int WINAPI IBSU_TakeResultImageManually
00543         (const int  handle);
00544
00568     int WINAPI IBSU_GetIBSM_ResultImageInfo
00569         (const int          handle,
00570         IBSM_FingerPosition fingerPosition,
00571         IBSM_ImageData*     pResultImage,
00572         IBSM_ImageData*     pSplitResultImage,
00573         int*                pSplitResultImageCount);
00574
00575
00592     int WINAPI IBSU_IsTouchedFinger
00593         (const int  handle,
00594         int*        pTouchInValue);
00612     int WINAPI IBSU_CheckWetFinger
00613         (const int              handle,
00614         const IBSU_ImageData    inImage);
00615
00635     int WINAPI IBSU_GetImageWidth
00636         (const int              handle,
00637         const IBSU_ImageData    inImage,
00638         int* Width_MM);
00639
```

```
00662     int WINAPI IBSU_ConvertImageToISOANSI
00663         (const int                  handle,
00664         const IBSM_ImageData*       image,
00665         const int                   imageCount,
00666         const IBSM_ImageFormat      imageFormat,
00667         const IBSM_StandardFormat   STDformat,
00668         IBSM_StandardFormatData*    pdata);
00669
00701     int WINAPI IBSU_GetContrast
00702         (const int  handle,
00703         int         *pContrastValue);
00704
00721     int WINAPI IBSU_SetContrast
00722         (const int  handle,
00723         const int   contrastValue);
00724
00740     int WINAPI IBSU_SetLEOperationMode
00741         (const int                  handle,
00742         const IBSU_LEOperationMode  leOperationMode);
00743
00759     int WINAPI IBSU_GetLEOperationMode
00760     (const int                  handle,
00761     IBSU_LEOperationMode* pLeOperationMode);
00762
00780     int WINAPI IBSU_GetOperableLEDs
00781         (const int      handle,
00782         IBSU_LedType    *pLedType,
00783         int             *pLedCount,
00784         DWORD           *pOperableLEDs);
00785
00803     int WINAPI IBSU_GetOperableLEDs
00804         (const int      handle,
00805         IBSU_LedType    *pLedType,
00806         int             *pLedCount,
00807         DWORD           *pOperableLEDs);
00808
00824     int WINAPI IBSU_GetLEDs
00825         (const int  handle,
00826         DWORD       *pActiveLEDs);
00827
00843     int WINAPI IBSU_SetLEDs
00844         (const int  handle,
00845         const DWORD activeLEDs);
00846
00847
00848
      /**************************************************************************************************
00849     * IBSU_GetOperableBeeper()
00850     *
00851     * @brief
00852     *     Get characteristics of operable Beeper on a device.
00853     *
00854     * @param
00855     *     handle         Device handle.
00856     * @param
00857     *     pBeeperType    Pointer to variable that will receive the type of Beeper.
00858     *
00859     * @return
00860     *           ::IBSU_STATUS_OK, if successful.\n
00861     *           Error code < 0, otherwise.  @see See error codes in 'IBScanUltimateApi_err.h'. \n
00862
      **************************************************************************************************
00863     */
00864     int WINAPI IBSU_GetOperableBeeper
00865         (const int      handle,
00866         IBSU_BeeperType *pBeeperType);
00867
00899     int WINAPI IBSU_SetBeeper
00900         (const int              handle,
00901         const IBSU_BeepPattern  beepPattern,
00902         const DWORD             soundTone,
00903         const DWORD             duration,
00904         const DWORD             reserved_1,
00905         const DWORD             reserved_2);
00906
00938     int WINAPI IBSU_BGetImage
00939     (const int                  handle,
00940         IBSU_ImageData* pImage,
00941         IBSU_ImageType* pImageType,
00942         IBSU_ImageData* pSplitImageArray,
00943         int* pSplitImageArrayCount,
00944         IBSU_FingerCountState* pFingerCountState,
00945         IBSU_FingerQualityState* pQualityArray,
00946         int* pQualityArrayCount);
00947
00984     int WINAPI IBSU_BGetImageEx
00985     (const int                  handle,
```

```
00986         int* pImageStatus,
00987         IBSU_ImageData* pImage,
00988         IBSU_ImageType* pImageType,
00989         int* pDetectedFingerCount,
00990         IBSU_ImageData* pSegmentImageArray,
00991         IBSU_SegmentPosition* pSegmentPositionArray,
00992         int* pSegmentImageArrayCount,
00993         IBSU_FingerCountState* pFingerCountState,
00994         IBSU_FingerQualityState* pQualityArray,
00995         int* pQualityArrayCount);
00996
01016     int WINAPI IBSU_BGetInitProgress
01017     (const int deviceIndex,
01018         BOOL* pIsComplete,
01019         int* pHandle,
01020         int* pProgressValue);
01021
01037     int WINAPI IBSU_BGetClearPlatenAtCapture
01038     (const int          handle,
01039         IBSU_PlatenState* pPlatenState);
01040
01058     int WINAPI IBSU_BGetRollingInfo
01059     (const int          handle,
01060         IBSU_RollingState* pRollingState,
01061         int* pRollingLineX);
01062
01085     int WINAPI IBSU_BGetRollingInfoEx
01086     (const int          handle,
01087         IBSU_RollingState* pRollingState,
01088         int* pRollingLineX,
01089         int* pRollingDirection,
01090         int* pRollingWidth);
01091
01126     int WINAPI IBSU_GenerateZoomOutImage
01127         (const IBSU_ImageData   inImage,
01128         BYTE* outImage,
01129         const int               outWidth,
01130         const int               outHeight,
01131         const BYTE              bkColor);
01132
01154     int WINAPI IBSU_GenerateZoomOutImageEx
01155         (const BYTE* pInImage,
01156         const int   inWidth,
01157         const int   inHeight,
01158         BYTE* outImage,
01159         const int   outWidth,
01160         const int   outHeight,
01161         const BYTE  bkColor);
01162
01184 #ifndef WINCE
01185     int WINAPI IBSU_SaveBitmapImage
01186         (LPCSTR         filePath,
01187         const BYTE* imgBuffer,
01188         const DWORD     width,
01189         const DWORD     height,
01190         const int       pitch,
01191         const double    resX,
01192         const double    resY);
01193
01194 #else  // UNICODE
01195     int WINAPI IBSU_SaveBitmapImageW
01196         (const wchar_t* filePath,
01197         const BYTE* imgBuffer,
01198         const DWORD     width,
01199         const DWORD     height,
01200         const int       pitch,
01201         const double    resX,
01202         const double    resY);
01203
01204 #define IBSU_SaveBitmapImage IBSU_SaveBitmapImageW
01205 #endif
01206
01236     int WINAPI IBSU_SaveBitmapMem
01237         (const BYTE* inImage,
01238         const DWORD             inWidth,
01239         const DWORD             inHeight,
01240         const int               inPitch,
01241         const double            inResX,
01242         const double            inResY,
01243         BYTE* outBitmapBuffer,
01244         const IBSU_ImageFormat  outImageFormat,
01245         const DWORD             outWidth,
01246         const DWORD             outHeight,
01247         const BYTE              bkColor);
01248
01287     int WINAPI IBSU_WSQEncodeMem
01288         (const BYTE* image,
```

```
01289         const int       width,
01290         const int       height,
01291         const int       pitch,
01292         const int       bitsPerPixel,
01293         const int       pixelPerInch,
01294         const double    bitRate,
01295         const char* commentText,
01296         BYTE** compressedData,
01297         int* compressedLength);
01298
01334 #ifndef WINCE
01335     int WINAPI IBSU_WSQEncodeToFile
01336         (LPCSTR         filePath,
01337         const BYTE* image,
01338         const int       width,
01339         const int       height,
01340         const int       pitch,
01341         const int       bitsPerPixel,
01342         const int       pixelPerInch,
01343         const double    bitRate,
01344         const char* commentText);
01345
01346 #else  // UNICODE
01347     int WINAPI IBSU_WSQEncodeToFileW
01348         (const wchar_t* filePath,
01349         const BYTE* image,
01350         const int       width,
01351         const int       height,
01352         const int       pitch,
01353         const int       bitsPerPixel,
01354         const int       pixelPerInch,
01355         const double    bitRate,
01356         const wchar_t* commentText);
01357
01358 #define IBSU_WSQEncodeToFile IBSU_WSQEncodeToFileW
01359 #endif
01360
01392     int WINAPI IBSU_WSQDecodeMem
01393         (const BYTE* compressedImage,
01394         const int   compressedLength,
01395         BYTE** decompressedImage,
01396         int* outWidth,
01397         int* outHeight,
01398         int* outPitch,
01399         int* outBitsPerPixel,
01400         int* outPixelPerInch);
01401
01431 #ifndef WINCE
01432     int WINAPI IBSU_WSQDecodeFromFile
01433         (LPCSTR filePath,
01434         BYTE** decompressedImage,
01435         int* outWidth,
01436         int* outHeight,
01437         int* outPitch,
01438         int* outBitsPerPixel,
01439         int* outPixelPerInch);
01440
01441 #else  // UNICODE
01442     int WINAPI IBSU_WSQDecodeFromFileW
01443         (const wchar_t* filePath,
01444         BYTE** decompressedImage,
01445         int* outWidth,
01446         int* outHeight,
01447         int* outPitch,
01448         int* outBitsPerPixel,
01449         int* outPixelPerInch);
01450
01451 #define IBSU_WSQDecodeFromFile IBSU_WSQDecodeFromFileW
01452 #endif
01453
01470     int WINAPI IBSU_FreeMemory
01471         (void* memblock);
01472
01501 #ifndef WINCE
01502     int WINAPI IBSU_SavePngImage
01503         (LPCSTR         filePath,
01504         const BYTE* image,
01505         const DWORD     width,
01506         const DWORD     height,
01507         const int       pitch,
01508         const double    resX,
01509         const double    resY
01510     );
01511
01512 #else  // UNICODE
01513     int WINAPI IBSU_SavePngImageW
01514         (const wchar_t* filePath,
```

```
01515         const BYTE* image,
01516         const DWORD     width,
01517         const DWORD     height,
01518         const int       pitch,
01519         const double    resX,
01520         const double    resY
01521     );
01522
01523 #define IBSU_SavePngImage IBSU_SavePngImageW
01524 #endif
01525
01556 #ifndef WINCE
01557     int WINAPI IBSU_SaveJP2Image
01558         (LPCSTR         filePath,
01559         const BYTE* image,
01560         const DWORD     width,
01561         const DWORD     height,
01562         const int       pitch,
01563         const double    resX,
01564         const double    resY,
01565         const int       fQuality
01566     );
01567
01568 #else  // UNICODE
01569     int WINAPI IBSU_SaveJP2ImageW
01570         (const wchar_t* filePath,
01571         const BYTE* image,
01572         const DWORD     width,
01573         const DWORD     height,
01574         const in        pitch,
01575         const double    resX,
01576         const double    resY,
01577         const int       fQuality
01578     );
01579
01580 #define IBSU_SaveJP2Image IBSU_SaveJP2ImageW
01581 #endif
01582
01604     int WINAPI IBSU_CombineImage
01605         (const IBSU_ImageData       inImage1,
01606         const IBSU_ImageData        inImage2,
01607         IBSU_CombineImageWhichHand  whichHand,
01608         IBSU_ImageData* outImage
01609     );
01610
01644     int WINAPI IBSU_CombineImageEx
01645         (const IBSU_ImageData       InImage1,
01646         const IBSU_ImageData        InImage2,
01647         IBSU_CombineImageWhichHand  WhichHand,
01648         IBSU_ImageData*             OutImage,
01649         IBSU_ImageData*             pSegmentImageArray,
01650         IBSU_SegmentPosition*       pSegmentPositionArray,
01651         int*                        pSegmentImageArrayCount
01652     );
01653
01689     int WINAPI IBSU_GenerateDisplayImage
01690     (const BYTE* pInImage,
01691         const int               inWidth,
01692         const int               inHeight,
01693         BYTE* outImage,
01694         const int               outWidth,
01695         const int               outHeight,
01696         const BYTE              outBkColor,
01697         const IBSU_ImageFormat  outFormat,
01698         const int               outQualityLevel,
01699         const BOOL              outVerticalFlip);
01700
01701
01736     int WINAPI IBSU_RemoveFingerImage
01737         (const int  handle,
01738         const DWORD fIndex);
01739
01766     int WINAPI IBSU_AddFingerImage
01767         (const int              handle,
01768         const IBSU_ImageData    image,
01769         const DWORD             fIndex,
01770         const IBSU_ImageType    imageType,
01771         const BOOL              flagForce);
01772
01799     int WINAPI IBSU_IsFingerDuplicated
01800         (const int              handle,
01801         const IBSU_ImageData    image,
01802         const DWORD             fIndex,
01803         const IBSU_ImageType    imageType,
01804         const int               securityLevel,
01805         DWORD*                  pMatchedPosition);
01806
```

```
01830     int WINAPI IBSU_IsValidFingerGeometry
01831         (const int              handle,
01832         const IBSU_ImageData    image,
01833         const DWORD             fIndex,
01834         const IBSU_ImageType    imageType,
01835         BOOL*                   pValid);
01836
01837
01873     int WINAPI IBSU_GetNFIQScore
01874         (const int  handle,
01875         const BYTE* imgBuffer,
01876         const DWORD width,
01877         const DWORD height,
01878         const BYTE  bitsPerPixel,
01879         int* pScore);
01880
01881     /*
01882
      ********************************************************************************************************
01883     * IBSU_GetNFIQScoreEx()
01884     *
01885     * DESCRIPTION:
01886     *     Calculate NFIQ score for an image. (Pitch argument added)
01887     *
01888     * ARGUMENTS:
01889     *     handle        Device handle.
01890     *     imgBuffer     Pointer to image buffer.
01891     *     width         Image width (in pixels).
01892     *     height        Image height (in pixels).
01893     *     pitch         Image pitch (in pixels).
01894     *     bitsPerPixel  Bits per pixel.
01895     *     pScore        Pointer to variable that will receive NFIQ score.
01896     *
01897     * RETURNS:
01898     *     IBSU_STATUS_OK, if successful.
01899     *     Error code < 0, otherwise.  See error codes in 'IBScanUltimateApi_err'.
01900
      ********************************************************************************************************
01901     */
01902     int WINAPI IBSU_GetNFIQScoreEx
01903         (const int  handle,
01904         const BYTE* imgBuffer,
01905         const DWORD width,
01906         const DWORD height,
01907         const int   pitch,
01908         const BYTE  bitsPerPixel,
01909         int* pScore);
01910
01944     int WINAPI IBSU_IsSpoofFingerDetected
01945         (const int              handle,
01946         const IBSU_ImageData    image,
01947         BOOL*                   pIsSpoof);
01948
01981         int WINAPI IBSU_SetEncryptionKey
01982             (const int                  handle,
01983             const unsigned char*        pEncyptionKey,
01984             const IBSU_EncryptionMode   encMode);
02017     int WINAPI IBSU_SetCustomerKey
02018         (const int          deviceIndex,
02019         const IBSU_HashType hashType,
02020         LPCSTR              pCustomerKey);
02048 #ifndef WINCE
02049     int WINAPI IBSU_GetSDKVersion
02050         (IBSU_SdkVersion* pVerinfo);
02051
02052 #else  // UNICODE
02053     int WINAPI IBSU_GetSDKVersionW
02054         (IBSU_SdkVersionW* pVerinfo);
02055
02056 #define IBSU_GetSDKVersion IBSU_GetSDKVersionW
02057 #endif
02058
02070     int WINAPI IBSU_UnloadLibrary();
02071
02090     int WINAPI IBSU_IsWritableDirectory
02091         (LPCSTR     dirpath,
02092         const BOOL  needCreateSubFolder);
02093
02112     int WINAPI IBSU_GetErrorString
02113         (const int  errorCode,
02114         LPSTR       errorString);
02115
02129     int WINAPI IBSU_EnableTraceLog
02130     (const BOOL on);
02131
02132
02168     int WINAPI IBSU_CreateClientWindow
```

```
02169         (const int      handle,
02170         const IBSU_HWND hWindow,
02171         const DWORD     left,
02172         const DWORD     top,
02173         const DWORD     right,
02174         const DWORD     bottom);
02175
02192     int WINAPI IBSU_DestroyClientWindow
02193         (const int  handle,
02194         const BOOL  clearExistingInfo);
02195
02215 #ifndef WINCE
02216     int WINAPI IBSU_GetClientWindowProperty
02217         (const int                          handle,
02218         const IBSU_ClientWindowPropertyId   propertyId,
02219         LPSTR                               propertyValue);
02220
02221 #else  // UNICODE
02222     int WINAPI IBSU_GetClientWindowPropertyW
02223         (const int                          handle,
02224         const IBSU_ClientWindowPropertyId   propertyId,
02225         wchar_t* propertyValue);
02226 #endif
02227
02246 #ifndef WINCE
02247     int WINAPI IBSU_SetClientDisplayProperty
02248         (const int                          handle,
02249         const IBSU_ClientWindowPropertyId   propertyId,
02250         LPCSTR                              propertyValue);
02251
02252 #else  // UNICODE
02253     int WINAPI IBSU_SetClientDisplayPropertyW
02254         (const int                          handle,
02255         const IBSU_ClientWindowPropertyId   propertyId,
02256         const wchar_t* propertyValue);
02257
02258 #define IBSU_SetClientDisplayProperty IBSU_SetClientDisplayPropertyW
02259 #endif
02260
02283 #ifndef WINCE
02284     int WINAPI IBSU_SetClientWindowOverlayText
02285         (const int  handle,
02286         const char* fontName,
02287         const int   fontSize,
02288         const BOOL  fontBold,
02289         const char* text,
02290         const int   posX,
02291         const int   posY,
02292         const DWORD textColor);
02293
02294 #else  // UNICODE
02295     int WINAPI IBSU_SetClientWindowOverlayTextW
02296         (const int      handle,
02297         const wchar_t*  fontName,
02298         const int       fontSize,
02299         const BOOL      fontBold,
02300         const wchar_t*  text,
02301         const int       posX,
02302         const int       posY,
02303         const DWORD     textColor);
02304
02305 #define IBSU_SetClientWindowOverlayText IBSU_SetClientWindowOverlayTextW
02306 #endif
02307
02308
02309
02310
02328     int WINAPI IBSU_ShowOverlayObject
02329         (const int  handle,
02330         const int   overlayHandle,
02331         const BOOL  show);
02332
02349     int WINAPI IBSU_ShowAllOverlayObject
02350         (const int  handle,
02351         const BOOL  show);
02352
02368     int WINAPI IBSU_RemoveOverlayObject
02369         (const int  handle,
02370         const int   overlayHandle);
02371
02386     int WINAPI IBSU_RemoveAllOverlayObject
02387         (const int  handle);
02388
02411 #ifndef WINCE
02412     int WINAPI IBSU_AddOverlayText
02413         (const int  handle,
02414         int*        pOverlayHandle,
```

```
02415         const char* fontName,
02416         const int   fontSize,
02417         const BOOL  fontBold,
02418         const char* text,
02419         const int   posX,
02420         const int   posY,
02421         const DWORD textColor);
02422
02423 #else  // UNICODE
02424     int WINAPI IBSU_AddOverlayTextW
02425         (const int      handle,
02426         int*            pOverlayHandle,
02427         const wchar_t*  fontName,
02428         const int       fontSize,
02429         const BOOL      fontBold,
02430         const wchar_t*  text,
02431         const int       posX,
02432         const int       posY,
02433         const DWORD     textColor);
02434
02435 #define IBSU_AddOverlayText IBSU_AddOverlayTextW
02436 #endif
02437
02460 #ifndef WINCE
02461     int WINAPI IBSU_ModifyOverlayText
02462         (const int  handle,
02463         const int   overlayHandle,
02464         const char* fontName,
02465         const int   fontSize,
02466         const BOOL  fontBold,
02467         const char* text,
02468         const int   posX,
02469         const int   posY,
02470         const DWORD textColor);
02471
02472 #else  // UNICODE
02473     int WINAPI IBSU_ModifyOverlayTextW
02474         (const int      handle,
02475         const int       overlayHandle,
02476         const wchar_t*  fontName,
02477         const int       fontSize,
02478         const BOOL      fontBold,
02479         const wchar_t*  text,
02480         const int       posX,
02481         const int       posY,
02482         const DWORD     textColor);
02483
02484 #define IBSU_ModifyOverlayText IBSU_ModifyOverlayTextW
02485 #endif
02486
02487
02517     int WINAPI IBSU_AddOverlayLine
02518     (const int      handle,
02519         int*        pOverlayHandle,
02520         const int   x1,
02521         const int   y1,
02522         const int   x2,
02523         const int   y2,
02524         const int   lineWidth,
02525         const DWORD lineColor);
02526
02548     int WINAPI IBSU_ModifyOverlayLine
02549     (const int      handle,
02550         const int   overlayHandle,
02551         const int   x1,
02552         const int   y1,
02553         const int   x2,
02554         const int   y2,
02555         const int   lineWidth,
02556         const DWORD lineColor);
02557
02583     int WINAPI IBSU_AddOverlayQuadrangle
02584     (const int  handle,
02585         int*        pOverlayHandle,
02586         const int   x1,
02587         const int   y1,
02588         const int   x2,
02589         const int   y2,
02590         const int   x3,
02591         const int   y3,
02592         const int   x4,
02593         const int   y4,
02594         const int   lineWidth,
02595         const DWORD lineColor);
02596
02622     int WINAPI IBSU_ModifyOverlayQuadrangle
02623         (const int  handle,
```

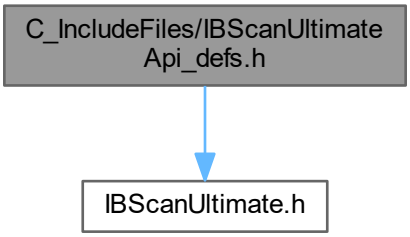
```
02624         const int   overlayHandle,
02625         const int   x1,
02626         const int   y1,
02627         const int   x2,
02628         const int   y2,
02629         const int   x3,
02630         const int   y3,
02631         const int   x4,
02632         const int   y4,
02633         const int   lineWidth,
02634         const DWORD lineColor);
02635
02663     int WINAPI IBSU_AddOverlayShape
02664         (const int                      handle,
02665         int*                            pOverlayHandle,
02666         const IBSU_OverlayShapePattern  shapePattern,
02667         const int                       x1,
02668         const int                       y1,
02669         const int                       x2,
02670         const int                       y2,
02671         const int                       lineWidth,
02672         const DWORD                     lineColor,
02673         const int                       reserved_1,
02674         const int                       reserved_2);
02675
02713     int WINAPI IBSU_ModifyOverlayShape
02714         (const int                      handle,
02715         const int                       overlayHandle,
02716         const IBSU_OverlayShapePattern  shapePattern,
02717         const int                       x1,
02718         const int                       y1,
02719         const int                       x2,
02720         const int                       y2,
02721         const int                       lineWidth,
02722         const DWORD                     lineColor,
02723         const int                       reserved_1,
02724         const int                       reserved_2);
02725
02726
02744     int WINAPI IBSU_RedrawClientWindow
02745         (const int  handle);
02746
02782     int WINAPI IBSU_RegisterCallbacks
02783         (const int          handle,
02784         const IBSU_Events   event,
02785         void*               pCallbackFunction,
02786         void*               pContext);
02787
02801     int WINAPI IBSU_ReleaseCallbacks
02802         (const int          handle,
02803         const IBSU_Events   events);
02804
02811 #ifdef __cplusplus
02812 } // extern "C"
02813 #endif
```

14.13 C_IncludeFiles/IBScanUltimateApi_defs.h File Reference

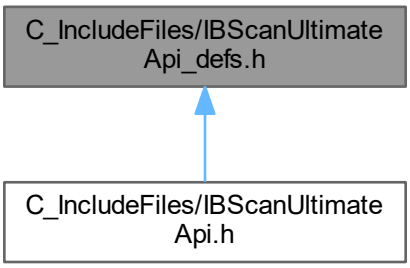
API structures and constants for IBScanUltimate.

```
#include "IBScanUltimate.h"
```

Include dependency graph for IBScanUltimateApi_defs.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct IBSU_SdkVersion
- struct IBSU_DeviceDesc
- struct Property_AlcorLink
- struct IBSU_SegmentPosition
- struct IBSM_ImageData
- struct IBSM_Template
- struct IBSM_StandardFormatData

Macros

- #define IBSU_HWND void *
- #define IBSU_RECT void *
- #define IBSU_MAX_STR_LEN 128
- #define IBSU_MIN_CONTRAST_VALUE 0

- #define IBSU_MAX_CONTRAST_VALUE 34
- #define IBSU_MAX_SEGMENT_COUNT 5
- #define IBSU_MAX_SEGMENT_QUALITY_COUNT 4
- #define IBSU_BMP_GRAY_HEADER_LEN 1078
- #define IBSU_BMP_RGB24_HEADER_LEN 54
- #define IBSU_BMP_RGB32_HEADER_LEN 54
- #define IBSU_OPTION_AUTO_CONTRAST 1
- #define IBSU_OPTION_AUTO_CAPTURE 2
- #define IBSU_OPTION_IGNORE_FINGER_COUNT 4
- #define IBSU_MAX_MINUTIAE_SIZE (255+2)
- #define IBSU_LED_NONE 0x00000000
- #define IBSU_LED_ALL 0xFFFFFFFF
- #define IBSU_LED_INIT_BLUE 0x00000001
- #define IBSU_LED_SCAN_GREEN 0x00000002
- #define IBSU_LED_SCAN_CURVE_RED 0x00000010
- #define IBSU_LED_SCAN_CURVE_GREEN 0x00000020
- #define IBSU_LED_SCAN_CURVE_BLUE 0x00000040
- #define IBSU_LED_F_BLINK_GREEN 0x10000000
- #define IBSU_LED_F_BLINK_RED 0x20000000
- #define IBSU_LED_F_LEFT_LITTLE_GREEN 0x01000000
- #define IBSU_LED_F_LEFT_LITTLE_RED 0x02000000
- #define IBSU_LED_F_LEFT_RING_GREEN 0x04000000
- #define IBSU_LED_F_LEFT_RING_RED 0x08000000
- #define IBSU_LED_F_LEFT_MIDDLE_GREEN 0x00100000
- #define IBSU_LED_F_LEFT_MIDDLE_RED 0x00200000
- #define IBSU_LED_F_LEFT_INDEX_GREEN 0x00400000
- #define IBSU_LED_F_LEFT_INDEX_RED 0x00800000
- #define IBSU_LED_F_LEFT_THUMB_GREEN 0x00010000
- #define IBSU_LED_F_LEFT_THUMB_RED 0x00020000
- #define IBSU_LED_F_RIGHT_THUMB_GREEN 0x00040000
- #define IBSU_LED_F_RIGHT_THUMB_RED 0x00080000
- #define IBSU_LED_F_RIGHT_INDEX_GREEN 0x00001000
- #define IBSU_LED_F_RIGHT_INDEX_RED 0x00002000
- #define IBSU_LED_F_RIGHT_MIDDLE_GREEN 0x00004000
- #define IBSU_LED_F_RIGHT_MIDDLE_RED 0x40000000
- #define IBSU_LED_F_RIGHT_RING_GREEN 0x00000100
- #define IBSU_LED_F_RIGHT_RING_RED 0x00000200
- #define IBSU_LED_F_RIGHT_LITTLE_GREEN 0x00000400
- #define IBSU_LED_F_RIGHT_LITTLE_RED 0x00000800
- #define IBSU_LED_F_PROGRESS_ROLL 0x00000010
- #define IBSU_LED_F_PROGRESS_LEFT_HAND 0x00000020
- #define IBSU_LED_F_PROGRESS_TWO_THUMB 0x00000040
- #define IBSU_LED_F_PROGRESS_RIGHT_HAND 0x00000080
- #define IBSU_FINGER_NONE 0x00000000
- #define IBSU_FINGER_LEFT_LITTLE 0x00000001
- #define IBSU_FINGER_LEFT_RING 0x00000002
- #define IBSU_FINGER_LEFT_MIDDLE 0x00000004
- #define IBSU_FINGER_LEFT_INDEX 0x00000008
- #define IBSU_FINGER_LEFT_THUMB 0x00000010
- #define IBSU_FINGER_RIGHT_THUMB 0x00000020
- #define IBSU_FINGER_RIGHT_INDEX 0x00000040
- #define IBSU_FINGER_RIGHT_MIDDLE 0x00000080
- #define IBSU_FINGER_RIGHT_RING 0x00000100
- #define IBSU_FINGER_RIGHT_LITTLE 0x00000200

- #define IBSU_FINGER_LEFT_HAND (IBSU_FINGER_LEFT_INDEX | IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_RING | IBSU_FINGER_LEFT_LITTLE)
- #define IBSU_FINGER_RIGHT_HAND (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE | IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)
- #define IBSU_FINGER_BOTH_THUMBS (IBSU_FINGER_RIGHT_THUMB | IBSU_FINGER_LEFT_THUMB)
- #define IBSU_FINGER_ALL (IBSU_FINGER_LEFT_HAND | IBSU_FINGER_RIGHT_HAND | IBSU_FINGER_BOTH_THUMBS
- #define IBSU_FINGER_LEFT_LITTLE_RING (IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING)
- #define IBSU_FINGER_LEFT_MIDDLE_INDEX (IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_INDEX)
- #define IBSU_FINGER_RIGHT_INDEX_MIDDLE (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE)
- #define IBSU_FINGER_RIGHT_RING_LITTLE (IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)
- #define PARALLEL_MAX_STR_LEN (32)

Typedefs

- typedef struct Property_AlcorLink * pProperty_AlcorLink
- typedef enum enumIBSU_BeeperType IBSU_BeeperType
- typedef enum enum_IBSM_FingerPosition IBSM_FingerPosition
- typedef enum enum_IBSM_TemplateVersion IBSM_TemplateVersion
- typedef void(CALLBACK * IBSU_Callback) (const int deviceHandle, void *pContext)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK, called when communication with a device is interrupted.
- typedef void(CALLBACK * IBSU_CallbackPreviewImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE, called when a preview image is available.
- typedef void(CALLBACK * IBSU_CallbackFingerCount) (const int deviceHandle, void *pContext, const IBSU_FingerCountState fingerCountState)

  Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT, called when the finger count changes.
- typedef void(CALLBACK * IBSU_CallbackFingerQuality) (const int deviceHandle, void *pContext, const IBSU_FingerQualityState *pQualityArray, const int qualityArrayCount)

  Callback for ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY, called when a finger quality changes.
- typedef void(CALLBACK * IBSU_CallbackDeviceCount) (const int detectedDevices, void *pContext)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT, called when the number of detected devices changes.
- typedef void(CALLBACK * IBSU_CallbackInitProgress) (const int deviceIndex, void *pContext, const int progressValue)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS, called when the initialization progress changes for a device.
- typedef void(CALLBACK * IBSU_CallbackTakingAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION, called for a rolled print acquisition when the rolling should begin.
- typedef void(CALLBACK * IBSU_CallbackCompleteAcquisition) (const int deviceHandle, void *pContext, const IBSU_ImageType imageType)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION, called for a rolled print acquisition when the rolling capture has completed.
- typedef void(CALLBACK * IBSU_CallbackResultImage) (const int deviceHandle, void *pContext, const IBSU_ImageData image, const IBSU_ImageType imageType, const IBSU_ImageData *pSplitImageArray, const int splitImageArrayCount)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE, called when the result image is available.
- typedef void(CALLBACK * IBSU_CallbackResultImageEx) (const int deviceHandle, void *pContext, const int imageStatus, const IBSU_ImageData image, const IBSU_ImageType imageType, const int detected↩ FingerCount, const int segmentImageArrayCount, const IBSU_ImageData *pSegmentImageArray, const IBSU_SegmentPosition *pSegmentPositionArray)

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

- typedef void(CALLBACK * IBSU_CallbackClearPlatenAtCapture) (const int deviceHandle, void *pContext, const IBSU_PlatenState platenState)

  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.
- typedef void(CALLBACK * IBSU_CallbackAsyncOpenDevice) (const int deviceIndex, void *pContext, const int deviceHandle, const int errorCode)

  Callback for ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE, called when asynchronous device initialization completes.
- typedef void(CALLBACK * IBSU_CallbackNotifyMessage) (const int deviceHandle, void *pContext, const int notifyMessage)

  Callback for ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE, called when a warning message is generated.
- typedef void(CALLBACK * IBSU_CallbackKeyButtons) (const int deviceHandle, void *pContext, const int pressedKeyButtons)

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

Enumerations

- enum IBSU_ImageType {
  ENUM_IBSU_TYPE_NONE , ENUM_IBSU_ROLL_SINGLE_FINGER , ENUM_IBSU_FLAT_SINGLE_FINGER
  , ENUM_IBSU_FLAT_TWO_FINGERS ,
  ENUM_IBSU_FLAT_FOUR_FINGERS , ENUM_IBSU_FLAT_THREE_FINGERS , ENUM_IBSU_FLAT_SINGLE_WRITERS_PA
  , ENUM_IBSU_FLAT_SINGLE_UPPER_PALM ,
  ENUM_IBSU_FLAT_SINGLE_LOWER_PALM }
- enum IBSU_ImageResolution { ENUM_IBSU_IMAGE_RESOLUTION_500 = 500 , ENUM_IBSU_IMAGE_RESOLUTION_1000 = 1000 }
- enum IBSU_PropertyId {
  ENUM_IBSU_PROPERTY_PRODUCT_ID , ENUM_IBSU_PROPERTY_SERIAL_NUMBER , ENUM_IBSU_PROPERTY_VEN
  , ENUM_IBSU_PROPERTY_IBIA_VENDOR_ID ,
  ENUM_IBSU_PROPERTY_IBIA_VERSION , ENUM_IBSU_PROPERTY_IBIA_DEVICE_ID , ENUM_IBSU_PROPERTY_FIRM
  , ENUM_IBSU_PROPERTY_REVISION ,
  ENUM_IBSU_PROPERTY_PRODUCTION_DATE , ENUM_IBSU_PROPERTY_SERVICE_DATE , ENUM_IBSU_PROPERTY_
  , ENUM_IBSU_PROPERTY_IMAGE_HEIGHT ,
  ENUM_IBSU_PROPERTY_IGNORE_FINGER_TIME , ENUM_IBSU_PROPERTY_RECOMMENDED_LEVEL
  , ENUM_IBSU_PROPERTY_POLLINGTIME_TO_BGETIMAGE , ENUM_IBSU_PROPERTY_ENABLE_POWER_SAVE_MODE
  ,
  ENUM_IBSU_PROPERTY_RETRY_WRONG_COMMUNICATION , ENUM_IBSU_PROPERTY_CAPTURE_TIMEOUT
  , ENUM_IBSU_PROPERTY_ROLL_MIN_WIDTH , ENUM_IBSU_PROPERTY_ROLL_MODE ,
  ENUM_IBSU_PROPERTY_ROLL_LEVEL , ENUM_IBSU_PROPERTY_CAPTURE_AREA_THRESHOLD ,
  ENUM_IBSU_PROPERTY_ENABLE_DECIMATION , ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_ON_RELEASE
  ,
  ENUM_IBSU_PROPERTY_DEVICE_INDEX , ENUM_IBSU_PROPERTY_DEVICE_ID , ENUM_IBSU_PROPERTY_SUPER_D
  , ENUM_IBSU_PROPERTY_MIN_CAPTURE_TIME_IN_SUPER_DRY_MODE ,
  ENUM_IBSU_PROPERTY_ROLLED_IMAGE_WIDTH , ENUM_IBSU_PROPERTY_ROLLED_IMAGE_HEIGHT
  , ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE , ENUM_IBSU_PROPERTY_ROLL_IMAGE_OVERRIDE
  ,
  ENUM_IBSU_PROPERTY_WARNING_MESSAGE_INVALID_AREA , ENUM_IBSU_PROPERTY_ENABLE_WET_FINGER_D
  , ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL , ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL_
  ,
  ENUM_IBSU_PROPERTY_START_POSITION_OF_ROLLING_AREA , ENUM_IBSU_PROPERTY_START_ROLL_WITHOUT_
  , ENUM_IBSU_PROPERTY_ENABLE_TOF , ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION ,
  ENUM_IBSU_PROPERTY_IS_SPOOF_SUPPORTED , ENUM_IBSU_PROPERTY_ENABLE_SPOOF ,
  ENUM_IBSU_PROPERTY_SPOOF_LEVEL , ENUM_IBSU_PROPERTY_VIEW_ENCRYPTION_IMAGE_MODE
  ,
  ENUM_IBSU_PROPERTY_FINGERPRINT_SEGMENTATION_MODE , ENUM_IBSU_PROPERTY_ROLL_METHOD

, ENUM_IBSU_PROPERTY_RENEWAL_OPPOSITE_IMGAE_LEVEL , ENUM_IBSU_PROPERTY_PREVIEW_IMAGE_QUAL
,
ENUM_IBSU_PROPERTY_ADAPTIVE_CAPTURE_MODE , ENUM_IBSU_PROPERTY_ENABLE_KOJAK_BEHAVIOR_2_6
, ENUM_IBSU_PROPERTY_DISABLE_SEGMENT_ROTATION , ENUM_IBSU_PROPERTY_DR_MODE_ZOOM_IN
,
ENUM_IBSU_PROPERTY_RESERVED_1 = 200 , ENUM_IBSU_PROPERTY_RESERVED_2 , ENUM_IBSU_PROPERTY_RE
, ENUM_IBSU_PROPERTY_RESERVED_IMAGE_PROCESS_THRESHOLD = 400 ,
ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TOF_FOR_ROLL , ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_B
, ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_ROLL ,
ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE ,
ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE_LEVEL , ENUM_IBSU_PROPERTY_RESERVED_E
, ENUM_IBSU_PROPERTY_RESERVED_SLIP_DETECTION_LEVEL , ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TF
,
ENUM_IBSU_PROPERTY_RESERVED_ENABLE_CBP_MODE , ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_WI
, ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_HEIGHT , ENUM_IBSU_PROPERTY_RESERVED_TFT_NOISE_RE
,
ENUM_IBSU_PROPERTY_RESERVED_LINE_BLACK_FILL , ENUM_IBSU_PROPERTY_RESERVED_RECALCILATE_BRIG
, ENUM_IBSU_PROPERTY_RESERVED_LINE_RESTORE , ENUM_IBSU_PROPERTY_RESERVED_SW_UNIFORMITY
,
ENUM_IBSU_PROPERTY_RESERVED_SET_ROLL_TEST_MODE }

- enum IBSU_ClientWindowPropertyId {
ENUM_IBSU_WINDOW_PROPERTY_BK_COLOR , ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE
, ENUM_IBSU_WINDOW_PROPERTY_DISP_INVALID_AREA , ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR
,
ENUM_IBSU_WINDOW_PROPERTY_LEFT_MARGIN , ENUM_IBSU_WINDOW_PROPERTY_TOP_MARGIN
, ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_WIDTH , ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTO
,
ENUM_IBSU_WINDOW_PROPERTY_KEEP_REDRAW_LAST_IMAGE , ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE
}
- enum IBSU_FingerCountState { ENUM_IBSU_FINGER_COUNT_OK , ENUM_IBSU_TOO_MANY_FINGERS
, ENUM_IBSU_TOO_FEW_FINGERS , ENUM_IBSU_NON_FINGER }
- enum IBSU_FingerQualityState {
ENUM_IBSU_FINGER_NOT_PRESENT , ENUM_IBSU_QUALITY_GOOD , ENUM_IBSU_QUALITY_FAIR
, ENUM_IBSU_QUALITY_POOR ,
ENUM_IBSU_QUALITY_INVALID_AREA_TOP , ENUM_IBSU_QUALITY_INVALID_AREA_LEFT ,
ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT , ENUM_IBSU_QUALITY_INVALID_AREA_BOTTOM }
- enum IBSU_LEOperationMode { ENUM_IBSU_LE_OPERATION_AUTO , ENUM_IBSU_LE_OPERATION_ON
, ENUM_IBSU_LE_OPERATION_OFF }
- enum IBSU_PlatenState { ENUM_IBSU_PLATEN_CLEARD , ENUM_IBSU_PLATEN_HAS_FINGERS }
- enum IBSU_Events {
ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT , ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK
, ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE , ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION
,
ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION , ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE
, ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY , ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT
,
ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS , ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE
, ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE , ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE
,
ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE_EX , ENUM_IBSU_ESSENTIAL_EVENT_KEYBUTTON
}
- enum IBSU_LedType { ENUM_IBSU_LED_TYPE_NONE , ENUM_IBSU_LED_TYPE_TSCAN , ENUM_IBSU_LED_TYPE_FSC
}
- enum IBSU_RollingState { ENUM_IBSU_ROLLING_NOT_PRESENT , ENUM_IBSU_ROLLING_TAKE_ACQUISITION
, ENUM_IBSU_ROLLING_COMPLETE_ACQUISITION , ENUM_IBSU_ROLLING_RESULT_IMAGE }
- enum IBSU_OverlayShapePattern { ENUM_IBSU_OVERLAY_SHAPE_RECTANGLE , ENUM_IBSU_OVERLAY_SHAPE_ELLI
, ENUM_IBSU_OVERLAY_SHAPE_CROSS , ENUM_IBSU_OVERLAY_SHAPE_ARROW }

- enum IBSU_CombineImageWhichHand { ENUM_IBSU_COMBINE_IMAGE_LEFT_HAND , ENUM_IBSU_COMBINE_IMAGE_
  }
- enum enumIBSU_BeeperType { ENUM_IBSU_BEEPER_TYPE_NONE , ENUM_IBSU_BEEPER_TYPE_MONOTONE
  }
- enum IBSU_BeepPattern { ENUM_IBSU_BEEP_PATTERN_GENERIC , ENUM_IBSU_BEEP_PATTERN_REPEAT
  }
- enum IBSU_EncryptionMode { ENUM_IBSU_ENCRYPTION_KEY_RANDOM , ENUM_IBSU_ENCRYPTION_KEY_CUSTOM
  , ENUM_IBSU_ENCRYPTION_KEY_DEFAULT }
- enum IBSM_ImageFormat {
  IBSM_IMG_FORMAT_NO_BIT_PACKING =0 , IBSM_IMG_FORMAT_BIT_PACKED , IBSM_IMG_FORMAT_WSQ
  , IBSM_IMG_FORMAT_JPEG_LOSSY ,
  IBSM_IMG_FORMAT_JPEG2000_LOSSY , IBSM_IMG_FORMAT_JPEG2000_LOSSLESS , IBSM_IMG_FORMAT_PNG
  , IBSM_IMG_FORMAT_UNKNOWN }
- enum IBSM_ImpressionType {
  IBSM_IMPRESSION_TYPE_LIVE_SCAN_PLAIN =0 , IBSM_IMPRESSION_TYPE_LIVE_SCAN_ROLLED ,
  IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PLAIN , IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_ROLLED
  ,
  IBSM_IMPRESSION_TYPE_LATENT_IMPRESSION , IBSM_IMPRESSION_TYPE_LATENT_TRACING ,
  IBSM_IMPRESSION_TYPE_LATENT_PHOTO , IBSM_IMPRESSION_TYPE_LATENT_LIFT ,
  IBSM_IMPRESSION_TYPE_LIVE_SCAN_SWIPE , IBSM_IMPRESSION_TYPE_LIVE_SCAN_VERTICAL_ROLL
  , IBSM_IMPRESSION_TYPE_LIVE_SCAN_PALM , IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PALM ,
  IBSM_IMPRESSION_TYPE_LATENT_PALM_IMPRESSION , IBSM_IMPRESSION_TYPE_LATENT_PALM_TRACING
  , IBSM_IMPRESSION_TYPE_LATENT_PALM_PHOTO , IBSM_IMPRESSION_TYPE_LATENT_PALM_LIFT
  ,
  IBSM_IMPRESSION_TYPE_LIVE_SCAN_OPTICAL_CONTRCTLESS_PLAIN =24 , IBSM_IMPRESSION_TYPE_OTHER
  =28 , IBSM_IMPRESSION_TYPE_UNKNOWN =29 }
- enum enum_IBSM_FingerPosition {
  IBSM_FINGER_POSITION_UNKNOWN =0 , IBSM_FINGER_POSITION_RIGHT_THUMB , IBSM_FINGER_POSITION_RIGH
  , 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
  ,
  IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE_AND_RING , IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_R
  , IBSM_FINGER_POSITION_UNKNOWN_HALF_PALM , IBSM_FINGER_POSITION_RIGHT_HALF_PALM
  ,
  IBSM_FINGER_POSITION_LEFT_HALF_PALM , IBSM_FINGER_POSITION_RIGHT_LOWER_HALF_PALM
  , IBSM_FINGER_POSITION_RIGHT_UPPER_HALF_PALM , IBSM_FINGER_POSITION_LEFT_LOWER_HALF_PALM
  ,

IBSM_FINGER_POSITION_LEFT_UPPER_HALF_PALM , IBSM_FINGER_POSITION_LEFT_CENTER_JOINT , IBSM_FINGER_POSITION_RIGHT_CENTER_JOINT , IBSM_FINGER_POSITION_LEFT_SIDE_JOINT , IBSM_FINGER_POSITION_RIGHT_SIDE_JOINT , IBSM_FINGER_POSITION_LEFT_ROLL_JOINT , IBSM_FINGER_POSITION_RIGHT_ROLL_JOINT , IBSM_FINGER_POSITION_LEFT_ROLL_UP , IBSM_FINGER_POSITION_RIGHT_ROLL_UP }

- enum IBSM_CaptureDeviceTechID { IBSM_CAPTURE_DEVICE_UNKNOWN_OR_UNSPECIFIED =0 , IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TIR , IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_DIRECT_VIEW_ON_PLATEN , IBSM_CAPTURE_DEVICE_WHITE_L , IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TIR , IBSM_CAPTURE_DEVICE_MONOCHROMATIC_ , IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TOUCHLESS , IBSM_CAPTURE_DEVICE_MONOCH , IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_DIRECT_VIEW_ON_PLATEN , IBSM_CAPTURE_DEVICE_M , IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TIR , IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_D , IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TOUCHLESS , IBSM_CAPTURE_DEVICE_ELECTRO_LUMINESC , IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_CAPACITIVE , IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_RF , IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_THEMAL , IBSM_CAPTURE_DEVICE_PRESSURE_SENSITIVE , IBSM_CAPTURE_DEVICE_ULTRASOUND , IBSM_CAPTURE_DEVICE_MECHANICAL , IBSM_CAPTURE_DEVICE_GLASS_FIBER }
- enum IBSM_CaptureDeviceTypeID { IBSM_CAPTURE_DEVICE_TYPE_ID_UNKNOWN = 0x0000 , IBSM_CAPTURE_DEVICE_TYPE_ID_CURVE = 0x1004 , IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON = 0x1005 , IBSM_CAPTURE_DEVICE_TYPE_ID_SHERLOCK = 0x1010 , IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON_MINI = 0x1020 , IBSM_CAPTURE_DEVICE_TYPE_ID_COLUMBO = 0x1100 , IBSM_CAPTURE_DEVICE_TYPE_ID_HOLMES = 0x1200 , IBSM_CAPTURE_DEVICE_TYPE_ID_KOJAK = 0x1300 , IBSM_CAPTURE_DEVICE_TYPE_ID_FIVE0 = 0x1500 , IBSM_CAPTURE_DEVICE_TYPE_ID_DANNO = 0x1600 , IBSM_CAPTURE_DEVICE_TYPE_ID_MANNIX = 0x1D00 }
- enum IBSM_CaptureDeviceVendorID { IBSM_CAPTURE_DEVICE_VENDOR_ID_UNREPORTED =0x0000 , IBSM_CAPTURE_DEVICE_VENDOR_INTEGRATED_BIOMETRICS =0x113F }
- enum IBSU_HashType { ENUM_IBSU_HASH_TYPE_SHA256 , ENUM_IBSU_HASH_TYPE_RESERVED }
- enum enum_IBSM_TemplateVersion { IBSM_TEMPLATE_VERSION_IBISDK_0 =0x00 , IBSM_TEMPLATE_VERSION_IBISDK_1 , IBSM_TEMPLATE_VERSION_IBI , IBSM_TEMPLATE_VERSION_IBISDK_3 , IBSM_TEMPLATE_VERSION_NEW_0 =0x10 }
- enum IBSM_StandardFormat { ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2005 , ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2005 , ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2011 , ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2011 , ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_378_2004 , ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_381_20 }

14.13.1 Detailed Description

API structures and constants for IBScanUltimate.

IBScanUltimateApi_defs.h

Author

Integrated Biometrics, LLC

Copyright

Definition in file IBScanUltimateApi_defs.h.

14.13.2 Macro Definition Documentation

14.13.2.1 IBSU_HWND

```
#define IBSU_HWND void *
```

Definition at line 206 of file IBScanUltimateApi_defs.h.

14.13.2.2 IBSU_RECT

```
#define IBSU_RECT void *
```

Definition at line 207 of file IBScanUltimateApi_defs.h.

14.14 IBScanUltimateApi_defs.h

Go to the documentation of this file.

```
00001
00198 #pragma once
00199
00200 #include "IBScanUltimate.h"
00201
00202 #ifdef _WINDOWS
00203 #define IBSU_HWND   HWND
00204 #define IBSU_RECT   RECT
00205 #else
00206 #define IBSU_HWND   void *
00207 #define IBSU_RECT   void *
00208 #endif
00209
00210 #ifdef __cplusplus
00211 extern "C" {
00212 #endif
00213
00222 #define IBSU_MAX_STR_LEN               128
00223
00225 #define IBSU_MIN_CONTRAST_VALUE          0
00226
00228 #define IBSU_MAX_CONTRAST_VALUE         34
00229
00231 #define IBSU_MAX_SEGMENT_COUNT           5
00233 #define IBSU_MAX_SEGMENT_QUALITY_COUNT   4
00235 #define IBSU_BMP_GRAY_HEADER_LEN      1078
00237 #define IBSU_BMP_RGB24_HEADER_LEN       54
00239 #define IBSU_BMP_RGB32_HEADER_LEN       54
00240
00242 #define IBSU_OPTION_AUTO_CONTRAST        1
00243 #define IBSU_OPTION_AUTO_CAPTURE         2
00244 #define IBSU_OPTION_IGNORE_FINGER_COUNT  4
00245
00247 #define IBSU_MAX_MINUTIAE_SIZE      (255+2)
00248
00249
00266 #define IBSU_LED_NONE              0x00000000
00268 #define IBSU_LED_ALL               0xFFFFFFFF
00270 #define IBSU_LED_INIT_BLUE         0x00000001
00272 #define IBSU_LED_SCAN_GREEN        0x00000002
00274 #define IBSU_LED_SCAN_CURVE_RED    0x00000010
00275 /* Green LED for Curve (TBN240). */
00276 #define IBSU_LED_SCAN_CURVE_GREEN  0x00000020
00278 #define IBSU_LED_SCAN_CURVE_BLUE   0x00000040
00295 #define IBSU_LED_F_BLINK_GREEN             0x10000000
00297 #define IBSU_LED_F_BLINK_RED               0x20000000
```

```
00299 #define IBSU_LED_F_LEFT_LITTLE_GREEN        0x01000000
00301 #define IBSU_LED_F_LEFT_LITTLE_RED          0x02000000
00303 #define IBSU_LED_F_LEFT_RING_GREEN          0x04000000
00305 #define IBSU_LED_F_LEFT_RING_RED            0x08000000
00307 #define IBSU_LED_F_LEFT_MIDDLE_GREEN        0x00100000
00309 #define IBSU_LED_F_LEFT_MIDDLE_RED          0x00200000
00311 #define IBSU_LED_F_LEFT_INDEX_GREEN         0x00400000
00313 #define IBSU_LED_F_LEFT_INDEX_RED           0x00800000
00315 #define IBSU_LED_F_LEFT_THUMB_GREEN         0x00010000
00317 #define IBSU_LED_F_LEFT_THUMB_RED           0x00020000
00319 #define IBSU_LED_F_RIGHT_THUMB_GREEN        0x00040000
00321 #define IBSU_LED_F_RIGHT_THUMB_RED          0x00080000
00323 #define IBSU_LED_F_RIGHT_INDEX_GREEN        0x00001000
00325 #define IBSU_LED_F_RIGHT_INDEX_RED          0x00002000
00327 #define IBSU_LED_F_RIGHT_MIDDLE_GREEN       0x00004000
00330 //#define IBSU_LED_F_RIGHT_MIDDLE_RED         0x00008000
00332 #define IBSU_LED_F_RIGHT_MIDDLE_RED         0x40000000
00334 #define IBSU_LED_F_RIGHT_RING_GREEN         0x00000100
00336 #define IBSU_LED_F_RIGHT_RING_RED           0x00000200
00338 #define IBSU_LED_F_RIGHT_LITTLE_GREEN       0x00000400
00340 #define IBSU_LED_F_RIGHT_LITTLE_RED         0x00000800
00342 #define IBSU_LED_F_PROGRESS_ROLL            0x00000010
00344 #define IBSU_LED_F_PROGRESS_LEFT_HAND       0x00000020
00346 #define IBSU_LED_F_PROGRESS_TWO_THUMB       0x00000040
00348 #define IBSU_LED_F_PROGRESS_RIGHT_HAND      0x00000080
00364 #define IBSU_FINGER_NONE                    0x00000000
00365 #define IBSU_FINGER_LEFT_LITTLE             0x00000001
00366 #define IBSU_FINGER_LEFT_RING               0x00000002
00367 #define IBSU_FINGER_LEFT_MIDDLE             0x00000004
00368 #define IBSU_FINGER_LEFT_INDEX              0x00000008
00369 #define IBSU_FINGER_LEFT_THUMB              0x00000010
00370 #define IBSU_FINGER_RIGHT_THUMB             0x00000020
00371 #define IBSU_FINGER_RIGHT_INDEX             0x00000040
00372 #define IBSU_FINGER_RIGHT_MIDDLE            0x00000080
00373 #define IBSU_FINGER_RIGHT_RING              0x00000100
00374 #define IBSU_FINGER_RIGHT_LITTLE            0x00000200
00375 #define IBSU_FINGER_LEFT_HAND               (IBSU_FINGER_LEFT_INDEX  | IBSU_FINGER_LEFT_MIDDLE  |
      IBSU_FINGER_LEFT_RING  | IBSU_FINGER_LEFT_LITTLE)
00376 #define IBSU_FINGER_RIGHT_HAND              (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE |
      IBSU_FINGER_RIGHT_RING | IBSU_FINGER_RIGHT_LITTLE)
00377 #define IBSU_FINGER_BOTH_THUMBS             (IBSU_FINGER_RIGHT_THUMB | IBSU_FINGER_LEFT_THUMB)
00378 #define IBSU_FINGER_ALL                     (IBSU_FINGER_LEFT_HAND   | IBSU_FINGER_RIGHT_HAND   |
      IBSU_FINGER_BOTH_THUMBS)
00379 #define IBSU_FINGER_LEFT_LITTLE_RING        (IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING)
00380 #define IBSU_FINGER_LEFT_MIDDLE_INDEX       (IBSU_FINGER_LEFT_MIDDLE | IBSU_FINGER_LEFT_INDEX)
00381 #define IBSU_FINGER_RIGHT_INDEX_MIDDLE      (IBSU_FINGER_RIGHT_INDEX | IBSU_FINGER_RIGHT_MIDDLE)
00382 #define IBSU_FINGER_RIGHT_RING_LITTLE       (IBSU_FINGER_RIGHT_RING  | IBSU_FINGER_RIGHT_LITTLE)
00398 #ifndef WINCE
00399 typedef struct
00400 {
00402     char Product[IBSU_MAX_STR_LEN];
00404     char File[IBSU_MAX_STR_LEN];
00405 }
00406 IBSU_SdkVersion;
00407
00408 #else  // UNICODE
00409 typedef struct
00410 {
00412     wchar_t Product[IBSU_MAX_STR_LEN];
00414     wchar_t File[IBSU_MAX_STR_LEN];
00415 }
00416 IBSU_SdkVersionW;
00417
00418 #define IBSU_SdkVersion IBSU_SdkVersionW
00419 #endif
00435 #ifndef WINCE
00436 typedef struct
00437 {
00439     char serialNumber[IBSU_MAX_STR_LEN];
00441     char productName[IBSU_MAX_STR_LEN];
00443     char interfaceType[IBSU_MAX_STR_LEN];
00445     char fwVersion[IBSU_MAX_STR_LEN];
00447     char devRevision[IBSU_MAX_STR_LEN];
00449     int  handle;
00451     BOOL IsHandleOpened;
00452 #ifdef __android__
00454     int  devID;
00455 #endif
00456     /* Check if device is locked. */
00457     BOOL IsDeviceLocked;
00458     /* CustomerString to display */
00459     char customerString[IBSU_MAX_STR_LEN];
00460 }
00461 IBSU_DeviceDesc;
00462
00463 #else  // UNICODE
00464 typedef struct
```

```
00465 {
00467     wchar_t serialNumber[IBSU_MAX_STR_LEN];
00469     wchar_t productName[IBSU_MAX_STR_LEN];
00471     wchar_t interfaceType[IBSU_MAX_STR_LEN];
00473     wchar_t fwVersion[IBSU_MAX_STR_LEN];
00475     wchar_t devRevision[IBSU_MAX_STR_LEN];
00477     int   handle;
00479     BOOL  IsHandleOpened;
00480 #ifdef __android__
00482     int  devID;
00483 #endif
00484     /* Check if device is locked. */
00485     BOOL IsDeviceLocked;
00486     /* CustomerString to display */
00487     wchar_t customerString[IBSU_MAX_STR_LEN];
00488 }
00489 IBSU_DeviceDescW;
00490
00491 #define IBSU_DeviceDesc IBSU_DeviceDescW
00492 #endif
00508 #define PARALLEL_MAX_STR_LEN    (32)
00509 typedef struct
00510 {
00511     char            cProductID[PARALLEL_MAX_STR_LEN];
00512     char            cSerialNumber[PARALLEL_MAX_STR_LEN];
00513     char            cVendorID[PARALLEL_MAX_STR_LEN];
00514     char            cIBIA_VendorID[PARALLEL_MAX_STR_LEN];
00515     char            cIBIA_Version[PARALLEL_MAX_STR_LEN];
00516     char            cIBIA_DeviceID[PARALLEL_MAX_STR_LEN];
00517     char            cFirmware[PARALLEL_MAX_STR_LEN];
00518     char            cDevRevision[PARALLEL_MAX_STR_LEN];
00519     char            cProductionDate[PARALLEL_MAX_STR_LEN];
00520     char            cServiceDate[PARALLEL_MAX_STR_LEN];
00521     char            cFPGA[PARALLEL_MAX_STR_LEN];
00522     char            cCMT1[PARALLEL_MAX_STR_LEN];
00523     char            cCMT2[PARALLEL_MAX_STR_LEN];
00524     char            cCMT3[PARALLEL_MAX_STR_LEN];
00525     char            cCMT4[PARALLEL_MAX_STR_LEN];
00526     unsigned short  idVendor;
00527     unsigned short  idProduct;
00528     char            cCalibrationData_str1[PARALLEL_MAX_STR_LEN];
00529     char            cCalibrationData_str2[PARALLEL_MAX_STR_LEN];
00530 } Property_AlcorLink, *pProperty_AlcorLink;
00546 typedef struct
00547 {
00549     short x1;
00551     short y1;
00553     short x2;
00555     short y2;
00557     short x3;
00559     short y3;
00561     short x4;
00563     short y4;
00564 }
00565 IBSU_SegmentPosition;
00583 typedef enum
00584 {
00586     ENUM_IBSU_TYPE_NONE,
00587
00589     ENUM_IBSU_ROLL_SINGLE_FINGER,
00590
00592     ENUM_IBSU_FLAT_SINGLE_FINGER,
00593
00595     ENUM_IBSU_FLAT_TWO_FINGERS,
00596
00598     ENUM_IBSU_FLAT_FOUR_FINGERS,
00599
00601     ENUM_IBSU_FLAT_THREE_FINGERS,
00602
00604     ENUM_IBSU_FLAT_SINGLE_WRITERS_PALM,
00605
00607     ENUM_IBSU_FLAT_SINGLE_UPPER_PALM,
00608
00610     ENUM_IBSU_FLAT_SINGLE_LOWER_PALM
00611 }
00612 IBSU_ImageType;
00629 typedef enum
00630 {
00632     ENUM_IBSU_IMAGE_RESOLUTION_500 = 500,
00633
00635     ENUM_IBSU_IMAGE_RESOLUTION_1000 = 1000
00636 }
00637 IBSU_ImageResolution;
00654 typedef enum
00655 {
00657     ENUM_IBSU_PROPERTY_PRODUCT_ID,
00658
```

```
00660     ENUM_IBSU_PROPERTY_SERIAL_NUMBER,
00661
00663     ENUM_IBSU_PROPERTY_VENDOR_ID,
00664
00666     ENUM_IBSU_PROPERTY_IBIA_VENDOR_ID,
00667
00669     ENUM_IBSU_PROPERTY_IBIA_VERSION,
00670
00672     ENUM_IBSU_PROPERTY_IBIA_DEVICE_ID,
00673
00675     ENUM_IBSU_PROPERTY_FIRMWARE,
00676
00678     ENUM_IBSU_PROPERTY_REVISION,
00679
00681     ENUM_IBSU_PROPERTY_PRODUCTION_DATE,
00682
00684     ENUM_IBSU_PROPERTY_SERVICE_DATE,
00685
00687     ENUM_IBSU_PROPERTY_IMAGE_WIDTH,
00688
00690     ENUM_IBSU_PROPERTY_IMAGE_HEIGHT,
00691
00695     ENUM_IBSU_PROPERTY_IGNORE_FINGER_TIME,
00696
00698     ENUM_IBSU_PROPERTY_RECOMMENDED_LEVEL,
00699
00701     ENUM_IBSU_PROPERTY_POLLINGTIME_TO_BGETIMAGE,
00702
00704     ENUM_IBSU_PROPERTY_ENABLE_POWER_SAVE_MODE,
00705
00708     ENUM_IBSU_PROPERTY_RETRY_WRONG_COMMUNICATION,
00709
00713     ENUM_IBSU_PROPERTY_CAPTURE_TIMEOUT,
00714
00717     ENUM_IBSU_PROPERTY_ROLL_MIN_WIDTH,
00718
00722     ENUM_IBSU_PROPERTY_ROLL_MODE,
00723
00728     ENUM_IBSU_PROPERTY_ROLL_LEVEL,
00729
00733     ENUM_IBSU_PROPERTY_CAPTURE_AREA_THRESHOLD,
00734
00737     ENUM_IBSU_PROPERTY_ENABLE_DECIMATION,
00738
00742     ENUM_IBSU_PROPERTY_ENABLE_CAPTURE_ON_RELEASE,
00743
00745     ENUM_IBSU_PROPERTY_DEVICE_INDEX,
00746
00748     ENUM_IBSU_PROPERTY_DEVICE_ID,
00749
00755     ENUM_IBSU_PROPERTY_SUPER_DRY_MODE,
00756
00761     ENUM_IBSU_PROPERTY_MIN_CAPTURE_TIME_IN_SUPER_DRY_MODE,
00762
00764     ENUM_IBSU_PROPERTY_ROLLED_IMAGE_WIDTH,
00765
00767     ENUM_IBSU_PROPERTY_ROLLED_IMAGE_HEIGHT,
00768
00771     ENUM_IBSU_PROPERTY_NO_PREVIEW_IMAGE,
00772
00775     ENUM_IBSU_PROPERTY_ROLL_IMAGE_OVERRIDE,
00776
00779     ENUM_IBSU_PROPERTY_WARNING_MESSAGE_INVALID_AREA,
00780
00783     ENUM_IBSU_PROPERTY_ENABLE_WET_FINGER_DETECT,
00784
00789     ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL,
00790
00795     ENUM_IBSU_PROPERTY_WET_FINGER_DETECT_LEVEL_THRESHOLD,
00796
00801     ENUM_IBSU_PROPERTY_START_POSITION_OF_ROLLING_AREA,
00802
00805     ENUM_IBSU_PROPERTY_START_ROLL_WITHOUT_LOCK,
00806
00809     ENUM_IBSU_PROPERTY_ENABLE_TOF,
00810
00813     ENUM_IBSU_PROPERTY_ENABLE_ENCRYPTION,
00814
00816     ENUM_IBSU_PROPERTY_IS_SPOOF_SUPPORTED,
00817
00820         ENUM_IBSU_PROPERTY_ENABLE_SPOOF,
00821
00836     ENUM_IBSU_PROPERTY_SPOOF_LEVEL,
00837
00840     ENUM_IBSU_PROPERTY_VIEW_ENCRYPTION_IMAGE_MODE,
00841
00844     ENUM_IBSU_PROPERTY_FINGERPRINT_SEGMENTATION_MODE,
```

```
00845
00848     ENUM_IBSU_PROPERTY_ROLL_METHOD,
00849
00857     ENUM_IBSU_PROPERTY_RENEWAL_OPPOSITE_IMGAE_LEVEL,
00858
00861     ENUM_IBSU_PROPERTY_PREVIEW_IMAGE_QUALITY_FOR_KOJAK,
00862
00865     ENUM_IBSU_PROPERTY_ADAPTIVE_CAPTURE_MODE,
00866
00869     ENUM_IBSU_PROPERTY_ENABLE_KOJAK_BEHAVIOR_2_6,
00870
00873     ENUM_IBSU_PROPERTY_DISABLE_SEGMENT_ROTATION,
00874
00878     ENUM_IBSU_PROPERTY_DR_MODE_ZOOM_IN,
00879
00881     ENUM_IBSU_PROPERTY_RESERVED_1 = 200,
00882     ENUM_IBSU_PROPERTY_RESERVED_2,
00883     ENUM_IBSU_PROPERTY_RESERVED_100,
00884
00892     ENUM_IBSU_PROPERTY_RESERVED_IMAGE_PROCESS_THRESHOLD = 400,
00893
00896     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TOF_FOR_ROLL,
00897
00900     ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_FLAT,
00901
00904     ENUM_IBSU_PROPERTY_RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_ROLL,
00905
00908     ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE,
00909
00914     ENUM_IBSU_PROPERTY_RESERVED_ENHANCED_RESULT_IMAGE_LEVEL,
00915
00918     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_SLIP_DETECTION,
00919
00922     ENUM_IBSU_PROPERTY_RESERVED_SLIP_DETECTION_LEVEL,
00923
00926     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_TRICK_CAPTURE,
00927
00930     ENUM_IBSU_PROPERTY_RESERVED_ENABLE_CBP_MODE,
00931
00932     /* Raw Image width value.  [Get only.] */
00933     ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_WIDTH,
00934
00935     /* Raw Image height value.  [Get only.] */
00936     ENUM_IBSU_PROPERTY_RESERVED_RAW_IMAGE_HEIGHT,
00937
00938
00939     ENUM_IBSU_PROPERTY_RESERVED_TFT_NOISE_REMOVAL,
00940
00941     ENUM_IBSU_PROPERTY_RESERVED_LINE_BLACK_FILL,
00942
00943     ENUM_IBSU_PROPERTY_RESERVED_RECALCILATE_BRIGHTNESS,
00944
00945     ENUM_IBSU_PROPERTY_RESERVED_LINE_RESTORE,
00946
00947     ENUM_IBSU_PROPERTY_RESERVED_SW_UNIFORMITY,
00948
00949     ENUM_IBSU_PROPERTY_RESERVED_SET_ROLL_TEST_MODE,
00950 }
00951 IBSU_PropertyId;
00968 typedef enum
00969 {
00972     ENUM_IBSU_WINDOW_PROPERTY_BK_COLOR,
00973
00976     ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE,
00977
00979     ENUM_IBSU_WINDOW_PROPERTY_DISP_INVALID_AREA,
00980
00982     ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR,
00983
00985     ENUM_IBSU_WINDOW_PROPERTY_LEFT_MARGIN,
00986
00988     ENUM_IBSU_WINDOW_PROPERTY_TOP_MARGIN,
00989
00992     ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_WIDTH,
00993
00995     ENUM_IBSU_WINDOW_PROPERTY_SCALE_FACTOR_EX,
00996
00999     ENUM_IBSU_WINDOW_PROPERTY_KEEP_REDRAW_LAST_IMAGE,
01000
01003     ENUM_IBSU_WINDOW_PROPERTY_ROLL_GUIDE_LINE_COLOR,
01004 }
01005 IBSU_ClientWindowPropertyId;
01021 typedef enum
01022 {
01023     ENUM_IBSU_FINGER_COUNT_OK,
01024     ENUM_IBSU_TOO_MANY_FINGERS,
01025     ENUM_IBSU_TOO_FEW_FINGERS,
```

```
01026     ENUM_IBSU_NON_FINGER
01027 }
01028 IBSU_FingerCountState;
01044 typedef enum
01045 {
01046     ENUM_IBSU_FINGER_NOT_PRESENT,
01047     ENUM_IBSU_QUALITY_GOOD,
01048     ENUM_IBSU_QUALITY_FAIR,
01049     ENUM_IBSU_QUALITY_POOR,
01051     ENUM_IBSU_QUALITY_INVALID_AREA_TOP,
01053     ENUM_IBSU_QUALITY_INVALID_AREA_LEFT,
01055     ENUM_IBSU_QUALITY_INVALID_AREA_RIGHT,
01057     ENUM_IBSU_QUALITY_INVALID_AREA_BOTTOM
01058 }
01059 IBSU_FingerQualityState;
01075 typedef enum
01076 {
01077     ENUM_IBSU_LE_OPERATION_AUTO,
01078     ENUM_IBSU_LE_OPERATION_ON,
01079     ENUM_IBSU_LE_OPERATION_OFF
01080 }
01081 IBSU_LEOperationMode;
01097 typedef enum
01098 {
01099     ENUM_IBSU_PLATEN_CLEARD,
01100     ENUM_IBSU_PLATEN_HAS_FINGERS
01101 }
01102 IBSU_PlatenState;
01118 typedef enum
01119 {
01121     ENUM_IBSU_ESSENTIAL_EVENT_DEVICE_COUNT,
01122
01124     ENUM_IBSU_ESSENTIAL_EVENT_COMMUNICATION_BREAK,
01125
01127     ENUM_IBSU_ESSENTIAL_EVENT_PREVIEW_IMAGE,
01128
01130     ENUM_IBSU_ESSENTIAL_EVENT_TAKING_ACQUISITION,
01131
01133     ENUM_IBSU_ESSENTIAL_EVENT_COMPLETE_ACQUISITION,
01134
01136     ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE,
01137
01139     ENUM_IBSU_OPTIONAL_EVENT_FINGER_QUALITY,
01140
01142     ENUM_IBSU_OPTIONAL_EVENT_FINGER_COUNT,
01143
01145     ENUM_IBSU_ESSENTIAL_EVENT_INIT_PROGRESS,
01146
01148     ENUM_IBSU_OPTIONAL_EVENT_CLEAR_PLATEN_AT_CAPTURE,
01149
01151     ENUM_IBSU_ESSENTIAL_EVENT_ASYNC_OPEN_DEVICE,
01152
01154     ENUM_IBSU_OPTIONAL_EVENT_NOTIFY_MESSAGE,
01155
01157     ENUM_IBSU_ESSENTIAL_EVENT_RESULT_IMAGE_EX,
01158
01160     ENUM_IBSU_ESSENTIAL_EVENT_KEYBUTTON
01161 }
01162 IBSU_Events;
01178 typedef enum
01179 {
01181     ENUM_IBSU_LED_TYPE_NONE,
01182
01184     ENUM_IBSU_LED_TYPE_TSCAN,
01185
01187     ENUM_IBSU_LED_TYPE_FSCAN
01188 }
01189 IBSU_LedType;
01205 typedef enum
01206 {
01207     ENUM_IBSU_ROLLING_NOT_PRESENT,
01208     ENUM_IBSU_ROLLING_TAKE_ACQUISITION,
01209     ENUM_IBSU_ROLLING_COMPLETE_ACQUISITION,
01210     ENUM_IBSU_ROLLING_RESULT_IMAGE
01211 }
01212 IBSU_RollingState;
01228 typedef enum
01229 {
01230     ENUM_IBSU_OVERLAY_SHAPE_RECTANGLE,
01231     ENUM_IBSU_OVERLAY_SHAPE_ELLIPSE,
01232     ENUM_IBSU_OVERLAY_SHAPE_CROSS,
01233     ENUM_IBSU_OVERLAY_SHAPE_ARROW
01234 }
01235 IBSU_OverlayShapePattern;
01251 typedef enum
01252 {
01253     ENUM_IBSU_COMBINE_IMAGE_LEFT_HAND,
```

```
01254     ENUM_IBSU_COMBINE_IMAGE_RIGHT_HAND
01255 }
01256 IBSU_CombineImageWhichHand;
01272 typedef enum enumIBSU_BeeperType
01273 {
01275     ENUM_IBSU_BEEPER_TYPE_NONE,
01276
01278     ENUM_IBSU_BEEPER_TYPE_MONOTONE,
01279 }
01280 IBSU_BeeperType;
01296 typedef enum
01297 {
01299     ENUM_IBSU_BEEP_PATTERN_GENERIC,
01300
01302     ENUM_IBSU_BEEP_PATTERN_REPEAT
01303 }
01304 IBSU_BeepPattern;
01320 typedef enum
01321 {
01323     ENUM_IBSU_ENCRYPTION_KEY_RANDOM,
01324
01326     ENUM_IBSU_ENCRYPTION_KEY_CUSTOM,
01327
01329     ENUM_IBSU_ENCRYPTION_KEY_DEFAULT
01330 }
01331 IBSU_EncryptionMode;
01347 typedef enum
01348 {
01349     IBSM_IMG_FORMAT_NO_BIT_PACKING=0,
01350     IBSM_IMG_FORMAT_BIT_PACKED,
01351     IBSM_IMG_FORMAT_WSQ,
01352     IBSM_IMG_FORMAT_JPEG_LOSSY,
01353     IBSM_IMG_FORMAT_JPEG2000_LOSSY,
01354     IBSM_IMG_FORMAT_JPEG2000_LOSSLESS,
01355     IBSM_IMG_FORMAT_PNG,
01356     IBSM_IMG_FORMAT_UNKNOWN
01357 }
01358 IBSM_ImageFormat;
01374 typedef enum
01375 {
01376     IBSM_IMPRESSION_TYPE_LIVE_SCAN_PLAIN=0,
01377     IBSM_IMPRESSION_TYPE_LIVE_SCAN_ROLLED,
01378     IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PLAIN,
01379     IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_ROLLED,
01380     IBSM_IMPRESSION_TYPE_LATENT_IMPRESSION,
01381     IBSM_IMPRESSION_TYPE_LATENT_TRACING,
01382     IBSM_IMPRESSION_TYPE_LATENT_PHOTO,
01383     IBSM_IMPRESSION_TYPE_LATENT_LIFT,
01384     IBSM_IMPRESSION_TYPE_LIVE_SCAN_SWIPE,
01385     IBSM_IMPRESSION_TYPE_LIVE_SCAN_VERTICAL_ROLL,
01386     IBSM_IMPRESSION_TYPE_LIVE_SCAN_PALM,
01387     IBSM_IMPRESSION_TYPE_NONLIVE_SCAN_PALM,
01388     IBSM_IMPRESSION_TYPE_LATENT_PALM_IMPRESSION,
01389     IBSM_IMPRESSION_TYPE_LATENT_PALM_TRACING,
01390     IBSM_IMPRESSION_TYPE_LATENT_PALM_PHOTO,
01391     IBSM_IMPRESSION_TYPE_LATENT_PALM_LIFT,
01392     IBSM_IMPRESSION_TYPE_LIVE_SCAN_OPTICAL_CONTRCTLESS_PLAIN=24,
01393     IBSM_IMPRESSION_TYPE_OTHER=28,
01394     IBSM_IMPRESSION_TYPE_UNKNOWN=29
01395 }
01396 IBSM_ImpressionType;
01412 typedef enum enum_IBSM_FingerPosition
01413 {
01414     IBSM_FINGER_POSITION_UNKNOWN=0,
01415     IBSM_FINGER_POSITION_RIGHT_THUMB,
01416     IBSM_FINGER_POSITION_RIGHT_INDEX_FINGER,
01417     IBSM_FINGER_POSITION_RIGHT_MIDDLE_FINGER,
01418     IBSM_FINGER_POSITION_RIGHT_RING_FINGER,
01419     IBSM_FINGER_POSITION_RIGHT_LITTLE_FINGER,
01420     IBSM_FINGER_POSITION_LEFT_THUMB,
01421     IBSM_FINGER_POSITION_LEFT_INDEX_FINGER,
01422     IBSM_FINGER_POSITION_LEFT_MIDDLE_FINGER,
01423     IBSM_FINGER_POSITION_LEFT_RING_FINGER,
01424     IBSM_FINGER_POSITION_LEFT_LITTLE_FINGER,
01425     IBSM_FINGER_POSITION_PLAIN_RIGHT_FOUR_FINGERS=13,
01426     IBSM_FINGER_POSITION_PLAIN_LEFT_FOUR_FINGERS,
01427     IBSM_FINGER_POSITION_PLAIN_THUMBS,
01428     IBSM_FINGER_POSITION_UNKNOWN_PALM=20,
01429     IBSM_FINGER_POSITION_RIGHT_FULL_PALM,
01430     IBSM_FINGER_POSITION_RIGHT_WRITERS_PALM,
01431     IBSM_FINGER_POSITION_LEFT_FULL_PALM,
01432     IBSM_FINGER_POSITION_LEFT_WRITERS_PALM,
01433     IBSM_FINGER_POSITION_RIGHT_LOWER_PALM,
01434     IBSM_FINGER_POSITION_RIGHT_UPPER_PALM,
01435     IBSM_FINGER_POSITION_LEFT_LOWER_PALM,
01436     IBSM_FINGER_POSITION_LEFT_UPPER_PALM,
01437     IBSM_FINGER_POSITION_RIGHT_OTHER,
```

```
01438     IBSM_FINGER_POSITION_LEFT_OTHER,
01439     IBSM_FINGER_POSITION_RIGHT_INTERDIGITAL,
01440     IBSM_FINGER_POSITION_RIGHT_THENAR,
01441     IBSM_FINGER_POSITION_RIGHT_HYPOTHENAR,
01442     IBSM_FINGER_POSITION_LEFT_INTERDIGITAL,
01443     IBSM_FINGER_POSITION_LEFT_THENAR,
01444     IBSM_FINGER_POSITION_LEFT_HYPOTHENAR,
01445     IBSM_FINGER_POSITION_RIGHT_INDEX_AND_MIDDLE=40,
01446     IBSM_FINGER_POSITION_RIGHT_MIDDLE_AND_RING,
01447     IBSM_FINGER_POSITION_RIGHT_RING_AND_LITTLE,
01448     IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE,
01449     IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_RING,
01450     IBSM_FINGER_POSITION_LEFT_RING_AND_LITTLE,
01451     IBSM_FINGER_POSITION_RIGHT_INDEX_AND_LEFT_INDEX,
01452     IBSM_FINGER_POSITION_RIGHT_INDEX_AND_MIDDLE_AND_RING,
01453     IBSM_FINGER_POSITION_RIGHT_MIDDLE_AND_RING_AND_LITTLE,
01454     IBSM_FINGER_POSITION_LEFT_INDEX_AND_MIDDLE_AND_RING,
01455     IBSM_FINGER_POSITION_LEFT_MIDDLE_AND_RING_AND_LITTLE,
01456
01457     // HALF PALM
01458     IBSM_FINGER_POSITION_UNKNOWN_HALF_PALM,
01459     IBSM_FINGER_POSITION_RIGHT_HALF_PALM,
01460     IBSM_FINGER_POSITION_LEFT_HALF_PALM,
01461     IBSM_FINGER_POSITION_RIGHT_LOWER_HALF_PALM,
01462     IBSM_FINGER_POSITION_RIGHT_UPPER_HALF_PALM,
01463     IBSM_FINGER_POSITION_LEFT_LOWER_HALF_PALM,
01464     IBSM_FINGER_POSITION_LEFT_UPPER_HALF_PALM,
01465
01466     IBSM_FINGER_POSITION_LEFT_CENTER_JOINT,
01467     IBSM_FINGER_POSITION_RIGHT_CENTER_JOINT,
01468     IBSM_FINGER_POSITION_LEFT_SIDE_JOINT,
01469     IBSM_FINGER_POSITION_RIGHT_SIDE_JOINT,
01470     IBSM_FINGER_POSITION_LEFT_ROLL_JOINT,
01471     IBSM_FINGER_POSITION_RIGHT_ROLL_JOINT,
01472     IBSM_FINGER_POSITION_LEFT_ROLL_UP,
01473     IBSM_FINGER_POSITION_RIGHT_ROLL_UP
01474 }
01475 IBSM_FingerPosition;
01491 typedef enum
01492 {
01493     IBSM_CAPTURE_DEVICE_UNKNOWN_OR_UNSPECIFIED=0,
01494     IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TIR,
01495     IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01496     IBSM_CAPTURE_DEVICE_WHITE_LIGHT_OPTICAL_TOUCHLESS,
01497     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TIR,
01498     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01499     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_VISIBLE_OPTICAL_TOUCHLESS,
01500     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_TIR,
01501     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01502     IBSM_CAPTURE_DEVICE_MONOCHROMATIC_IR_OPTICAL_TOUCHLESS,
01503     IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TIR,
01504     IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_DIRECT_VIEW_ON_PLATEN,
01505     IBSM_CAPTURE_DEVICE_MULTISPECTRAL_OPTICAL_TOUCHLESS,
01506     IBSM_CAPTURE_DEVICE_ELECTRO_LUMINESCENT,
01507     IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_CAPACITIVE,
01508     IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_RF,
01509     IBSM_CAPTURE_DEVICE_SEMICONDUCTOR_THEMAL,
01510     IBSM_CAPTURE_DEVICE_PRESSURE_SENSITIVE,
01511     IBSM_CAPTURE_DEVICE_ULTRASOUND,
01512     IBSM_CAPTURE_DEVICE_MECHANICAL,
01513     IBSM_CAPTURE_DEVICE_GLASS_FIBER
01514 }
01515 IBSM_CaptureDeviceTechID;
01531 typedef enum
01532 {
01533     IBSM_CAPTURE_DEVICE_TYPE_ID_UNKNOWN = 0x0000,
01534     IBSM_CAPTURE_DEVICE_TYPE_ID_CURVE = 0x1004,
01535     IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON = 0x1005,
01536     IBSM_CAPTURE_DEVICE_TYPE_ID_SHERLOCK = 0x1010,
01537     IBSM_CAPTURE_DEVICE_TYPE_ID_WATSON_MINI = 0x1020,
01538     IBSM_CAPTURE_DEVICE_TYPE_ID_COLUMBO = 0x1100,
01539     IBSM_CAPTURE_DEVICE_TYPE_ID_HOLMES = 0x1200,
01540     IBSM_CAPTURE_DEVICE_TYPE_ID_KOJAK = 0x1300,
01541     IBSM_CAPTURE_DEVICE_TYPE_ID_FIVE0 = 0x1500,
01542     IBSM_CAPTURE_DEVICE_TYPE_ID_DANNO = 0x1600,
01543     IBSM_CAPTURE_DEVICE_TYPE_ID_MANNIX = 0x1D00,
01544 }
01545 IBSM_CaptureDeviceTypeID;
01561 typedef enum
01562 {
01563     IBSM_CAPTURE_DEVICE_VENDOR_ID_UNREPORTED=0x0000,
01564     IBSM_CAPTURE_DEVICE_VENDOR_INTEGRATED_BIOMETRICS=0x113F
01565 }
01566 IBSM_CaptureDeviceVendorID;
01582 typedef enum
01583 {
01585     ENUM_IBSU_HASH_TYPE_SHA256,
```

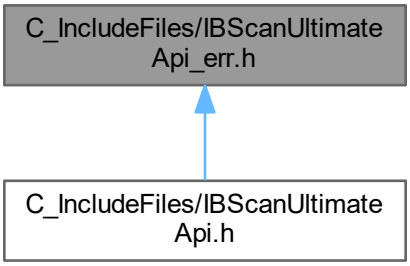
```
01586
01588     ENUM_IBSU_HASH_TYPE_RESERVED
01589 }
01590 IBSU_HashType;
01606 typedef struct
01607 {
01608     IBSM_ImageFormat         ImageFormat;
01609     IBSM_ImpressionType      ImpressionType;
01610     IBSM_FingerPosition      FingerPosition;
01611     IBSM_CaptureDeviceTechID CaptureDeviceTechID;
01612     unsigned short           CaptureDeviceVendorID;
01613     unsigned short           CaptureDeviceTypeID;
01614     unsigned short           ScanSamplingX;
01615     unsigned short           ScanSamplingY;
01616     unsigned short           ImageSamplingX;
01617     unsigned short           ImageSamplingY;
01618     unsigned short           ImageSizeX;
01619     unsigned short           ImageSizeY;
01620     unsigned char            ScaleUnit;
01621     unsigned char            BitDepth;
01622     unsigned int             ImageDataLength;
01623     void                     *ImageData;
01624 }
01625 IBSM_ImageData;
01641 typedef enum enum_IBSM_TemplateVersion
01642 {
01644     IBSM_TEMPLATE_VERSION_IBISDK_0=0x00,
01646     IBSM_TEMPLATE_VERSION_IBISDK_1,
01648     IBSM_TEMPLATE_VERSION_IBISDK_2,
01650     IBSM_TEMPLATE_VERSION_IBISDK_3,
01652     IBSM_TEMPLATE_VERSION_NEW_0=0x10
01653 }
01654 IBSM_TemplateVersion;
01670 typedef struct
01671 {
01672     IBSM_TemplateVersion        Version;
01673     unsigned int                FingerPosition;
01674     IBSM_ImpressionType         ImpressionType;
01675     IBSM_CaptureDeviceTechID    CaptureDeviceTechID;
01677     unsigned short              CaptureDeviceVendorID;
01679     unsigned short              CaptureDeviceTypeID;
01681     unsigned short              ImageSamplingX;
01683     unsigned short              ImageSamplingY;
01684     unsigned short              ImageSizeX;
01685     unsigned short              ImageSizeY;
01686     unsigned int                Minutiae[IBSU_MAX_MINUTIAE_SIZE];
01687     unsigned int                Reserved;
01688 }
01689 IBSM_Template;
01705 typedef enum
01706 {
01708     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2005,
01709
01711     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2005,
01712
01714     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_2_2011,
01715
01717     ENUM_IBSM_STANDARD_FORMAT_ISO_19794_4_2011,
01718
01720     ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_378_2004,
01721
01723     ENUM_IBSM_STANDARD_FORMAT_ANSI_INCITS_381_2004,
01724 }
01725 IBSM_StandardFormat;
01741 typedef struct
01742 {
01746     void                    *Data;
01747
01749     unsigned long           DataLength;
01750
01752     IBSM_StandardFormat     Format;
01753 }
01754 IBSM_StandardFormatData;
01784 typedef void (CALLBACK *IBSU_Callback)
01785     (const int deviceHandle,
01786         void     *pContext);
01787
01805 typedef void (CALLBACK *IBSU_CallbackPreviewImage)
01806     (const int            deviceHandle,
01807         void                *pContext,
01808         const IBSU_ImageData image);
01809
01825 typedef void (CALLBACK *IBSU_CallbackFingerCount)
01826     (const int                   deviceHandle,
01827         void                      *pContext,
01828         const IBSU_FingerCountState fingerCountState);
01829
```

```
01847 typedef void (CALLBACK *IBSU_CallbackFingerQuality)
01848     (const int                      deviceHandle,
01849         void                          *pContext,
01850         const IBSU_FingerQualityState *pQualityArray,
01851         const int                      qualityArrayCount);
01852
01867 typedef void (CALLBACK *IBSU_CallbackDeviceCount)
01868     (const int detectedDevices,
01869         void     *pContext);
01870
01887 typedef void (CALLBACK *IBSU_CallbackInitProgress)
01888     (const int deviceIndex,
01889         void     *pContext,
01890         const int progressValue);
01891
01908 typedef void (CALLBACK *IBSU_CallbackTakingAcquisition)
01909     (const int            deviceHandle,
01910         void               *pContext,
01911         const IBSU_ImageType imageType);
01912
01929 typedef void (CALLBACK *IBSU_CallbackCompleteAcquisition)
01930     (const int            deviceHandle,
01931         void               *pContext,
01932         const IBSU_ImageType imageType);
01933
01961 typedef void (CALLBACK *IBSU_CallbackResultImage)
01962     (const int             deviceHandle,
01963         void                *pContext,
01964         const IBSU_ImageData  image,
01965         const IBSU_ImageType  imageType,
01966         const IBSU_ImageData *pSplitImageArray,
01967         const int             splitImageArrayCount);
01968
02004 typedef void (CALLBACK *IBSU_CallbackResultImageEx)
02005     (const int                    deviceHandle,
02006         void                        *pContext,
02007         const int                    imageStatus,
02008         const IBSU_ImageData         image,
02009         const IBSU_ImageType         imageType,
02010         const int                    detectedFingerCount,
02011         const int                    segmentImageArrayCount,
02012         const IBSU_ImageData         *pSegmentImageArray,
02013         const IBSU_SegmentPosition   *pSegmentPositionArray);
02014
02031 typedef void (CALLBACK *IBSU_CallbackClearPlatenAtCapture)
02032     (const int              deviceHandle,
02033         void                 *pContext,
02034         const IBSU_PlatenState platenState);
02035
02054 typedef void (CALLBACK *IBSU_CallbackAsyncOpenDevice)
02055     (const int deviceIndex,
02056         void     *pContext,
02057         const int deviceHandle,
02058         const int errorCode);
02059
02076 typedef void (CALLBACK *IBSU_CallbackNotifyMessage)
02077     (const int deviceHandle,
02078         void     *pContext,
02079         const int notifyMessage);
02080
02096 typedef void (CALLBACK *IBSU_CallbackKeyButtons)
02097     (const int deviceHandle,
02098         void     *pContext,
02099         const int pressedKeyButtons);
02100
02108 #ifdef __cplusplus
02109 } // extern "C"
02110 #endif
```

14.15 C_IncludeFiles/IBScanUltimateApi_err.h File Reference

Error codes for IBScanUltimate.

This graph shows which files directly or indirectly include this file:



Macros

- #define IBSU_STATUS_OK 0
- #define IBSU_ERR_INVALID_PARAM_VALUE -1
- #define IBSU_ERR_MEM_ALLOC -2
- #define IBSU_ERR_NOT_SUPPORTED -3
- #define IBSU_ERR_FILE_OPEN -4
- #define IBSU_ERR_FILE_READ -5
- #define IBSU_ERR_RESOURCE_LOCKED -6
- #define IBSU_ERR_MISSING_RESOURCE -7
- #define IBSU_ERR_INVALID_ACCESS_POINTER -8
- #define IBSU_ERR_THREAD_CREATE -9
- #define IBSU_ERR_COMMAND_FAILED -10
- #define IBSU_ERR_LIBRARY_UNLOAD_FAILED -11
- #define IBSU_ERR_CHANNEL_IO_COMMAND_FAILED -100
- #define IBSU_ERR_CHANNEL_IO_READ_FAILED -101
- #define IBSU_ERR_CHANNEL_IO_WRITE_FAILED -102
- #define IBSU_ERR_CHANNEL_IO_READ_TIMEOUT -103
- #define IBSU_ERR_CHANNEL_IO_WRITE_TIMEOUT -104
- #define IBSU_ERR_CHANNEL_IO_UNEXPECTED_FAILED -105
- #define IBSU_ERR_CHANNEL_IO_INVALID_HANDLE -106
- #define IBSU_ERR_CHANNEL_IO_WRONG_PIPE_INDEX -107
- #define IBSU_ERR_DEVICE_IO -200
- #define IBSU_ERR_DEVICE_NOT_FOUND -201
- #define IBSU_ERR_DEVICE_NOT_MATCHED -202
- #define IBSU_ERR_DEVICE_ACTIVE -203
- #define IBSU_ERR_DEVICE_NOT_INITIALIZED -204
- #define IBSU_ERR_DEVICE_INVALID_STATE -205
- #define IBSU_ERR_DEVICE_BUSY -206
- #define IBSU_ERR_DEVICE_NOT_SUPPORTED_FEATURE -207
- #define IBSU_ERR_INVALID_LICENSE -208
- #define IBSU_ERR_USB20_REQUIRED -209
- #define IBSU_ERR_DEVICE_ENABLED_POWER_SAVE_MODE -210
- #define IBSU_ERR_DEVICE_NEED_UPDATE_FIRMWARE -211
- #define IBSU_ERR_DEVICE_NEED_CALIBRATE_TOF -212
- #define IBSU_ERR_DEVICE_INVALID_CALIBRATION_DATA -213

- #define IBSU_ERR_DEVICE_HIGHER_SDK_REQUIRED -214
- #define IBSU_ERR_DEVICE_LOCK_INVALID_BUFF -215
- #define IBSU_ERR_DEVICE_LOCK_INFO_EMPTY -216
- #define IBSU_ERR_DEVICE_LOCK_INFO_NOT_MATCHED -217
- #define IBSU_ERR_DEVICE_LOCK_INVALID_CHECKSUM -218
- #define IBSU_ERR_DEVICE_LOCK_INVALID_KEY -219
- #define IBSU_ERR_DEVICE_LOCK_LOCKED -220
- #define IBSU_ERR_DEVICE_LOCK_ILLEGAL_DEVICE -221
- #define IBSU_ERR_DEVICE_LOCK_INVALID_SERIAL_FORMAT -222
- #define IBSU_ERR_CAPTURE_COMMAND_FAILED -300
- #define IBSU_ERR_CAPTURE_STOP -301
- #define IBSU_ERR_CAPTURE_TIMEOUT -302
- #define IBSU_ERR_CAPTURE_STILL_RUNNING -303
- #define IBSU_ERR_CAPTURE_NOT_RUNNING -304
- #define IBSU_ERR_CAPTURE_INVALID_MODE -305
- #define IBSU_ERR_CAPTURE_ALGORITHM -306
- #define IBSU_ERR_CAPTURE_ROLLING -307
- #define IBSU_ERR_CAPTURE_ROLLING_TIMEOUT -308
- #define IBSU_ERR_CLIENT_WINDOW -400
- #define IBSU_ERR_CLIENT_WINDOW_NOT_CREATE -401
- #define IBSU_ERR_INVALID_OVERLAY_HANDLE -402
- #define IBSU_ERR_NBIS_NFIQ_FAILED -500
- #define IBSU_ERR_NBIS_WSQ_ENCODE_FAILED -501
- #define IBSU_ERR_NBIS_WSQ_DECODE_FAILED -502
- #define IBSU_ERR_NBIS_PNG_ENCODE_FAILED -503
- #define IBSU_ERR_NBIS_JP2_ENCODE_FAILED -504
- #define IBSU_ERR_DUPLICATE_EXTRACTION_FAILED -600
- #define IBSU_ERR_DUPLICATE_ALREADY_USED -601
- #define IBSU_ERR_DUPLICATE_SEGMENTATION_FAILED -602
- #define IBSU_ERR_DUPLICATE_MATCHING_FAILED -603
- #define IBSU_ERR_PAD_PROPERTY_DISABLED -700
- #define IBSU_ERR_INCORRECT_STANDARD_FORMAT -800
- #define IBSU_WRN_CHANNEL_IO_FRAME_MISSING 100
- #define IBSU_WRN_CHANNEL_IO_CAMERA_WRONG 101
- #define IBSU_WRN_CHANNEL_IO_SLEEP_STATUS 102
- #define IBSU_WRN_OUTDATED_FIRMWARE 200
- #define IBSU_WRN_ALREADY_INITIALIZED 201
- #define IBSU_WRN_API_DEPRECATED 202
- #define IBSU_WRN_ALREADY_ENHANCED_IMAGE 203
- #define IBSU_WRN_BGET_IMAGE 300
- #define IBSU_WRN_ROLLING_NOT_RUNNING 301
- #define IBSU_WRN_NO_FINGER 302
- #define IBSU_WRN_INCORRECT_FINGERS 303
- #define IBSU_WRN_EMPTY_IBSM_RESULT_IMAGE 400
- #define IBSU_WRN_INVALID_BRIGHTNESS_FINGERS 600
- #define IBSU_WRN_WET_FINGERS 601
- #define IBSU_WRN_MULTIPLE_FINGERS_DURING_ROLL 602
- #define IBSU_WRN_SPOOF_DETECTED 603
- #define IBSU_WRN_ROLLING_SLIP_DETECTED 604
- #define IBSU_WRN_SPOOF_INIT_FAILED 605
- #define IBSU_WRN_MATCHER_NO_MATCH 700
- #define IBSU_WRN_MATCHER_ALREADY_REGISTERED 701
- #define IBSU_WRN_ROLLING_SMEAR 304
- #define IBSU_WRN_ROLLING_SHIFTED_HORIZONTALLY (IBSU_WRN_ROLLING_SMEAR | 1)
- #define IBSU_WRN_ROLLING_SHIFTED_VERTICALLY (IBSU_WRN_ROLLING_SMEAR | 2)

- #define IBSU_WRN_QUALITY_INVALID_AREA 512
- #define IBSU_WRN_QUALITY_INVALID_AREA_HORIZONTALLY (IBSU_WRN_QUALITY_INVALID_AREA | 1)
- #define IBSU_WRN_QUALITY_INVALID_AREA_VERTICALLY (IBSU_WRN_QUALITY_INVALID_AREA | 2)

14.15.1 Detailed Description

Error codes for IBScanUltimate.

IBScanUltimateApi_err.h

Author

Integrated Biometrics, LLC

Copyright

Definition in file IBScanUltimateApi_err.h.

14.16 IBScanUltimateApi_err.h

Go to the documentation of this file.

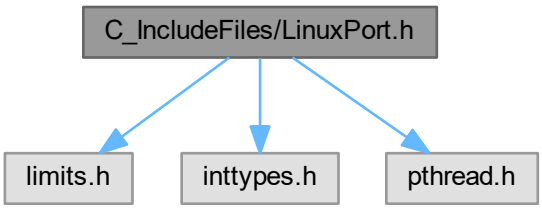
```
00001
00085 #pragma once
00086
00095 #define IBSU_STATUS_OK                                0
00097 #define IBSU_ERR_INVALID_PARAM_VALUE                 -1
00099 #define IBSU_ERR_MEM_ALLOC                           -2
00101 #define IBSU_ERR_NOT_SUPPORTED                       -3
00103 #define IBSU_ERR_FILE_OPEN                           -4
00105 #define IBSU_ERR_FILE_READ                           -5
00107 #define IBSU_ERR_RESOURCE_LOCKED                     -6
00109 #define IBSU_ERR_MISSING_RESOURCE                    -7
00111 #define IBSU_ERR_INVALID_ACCESS_POINTER              -8
00113 #define IBSU_ERR_THREAD_CREATE                       -9
00115 #define IBSU_ERR_COMMAND_FAILED                     -10
00117 #define IBSU_ERR_LIBRARY_UNLOAD_FAILED              -11
00132 #define IBSU_ERR_CHANNEL_IO_COMMAND_FAILED          -100
00134 #define IBSU_ERR_CHANNEL_IO_READ_FAILED             -101
00136 #define IBSU_ERR_CHANNEL_IO_WRITE_FAILED            -102
00138 #define IBSU_ERR_CHANNEL_IO_READ_TIMEOUT            -103
00140 #define IBSU_ERR_CHANNEL_IO_WRITE_TIMEOUT           -104
00142 #define IBSU_ERR_CHANNEL_IO_UNEXPECTED_FAILED       -105
00144 #define IBSU_ERR_CHANNEL_IO_INVALID_HANDLE          -106
00146 #define IBSU_ERR_CHANNEL_IO_WRONG_PIPE_INDEX        -107
00147
00162 #define IBSU_ERR_DEVICE_IO                          -200
00164 #define IBSU_ERR_DEVICE_NOT_FOUND                   -201
00166 #define IBSU_ERR_DEVICE_NOT_MATCHED                 -202
00168 #define IBSU_ERR_DEVICE_ACTIVE                      -203
00170 #define IBSU_ERR_DEVICE_NOT_INITIALIZED             -204
00172 #define IBSU_ERR_DEVICE_INVALID_STATE               -205
00174 #define IBSU_ERR_DEVICE_BUSY                        -206
00176 #define IBSU_ERR_DEVICE_NOT_SUPPORTED_FEATURE       -207
00178 #define IBSU_ERR_INVALID_LICENSE                    -208
00180 #define IBSU_ERR_USB20_REQUIRED                     -209
00182 #define IBSU_ERR_DEVICE_ENABLED_POWER_SAVE_MODE     -210
00184 #define IBSU_ERR_DEVICE_NEED_UPDATE_FIRMWARE        -211
00186 #define IBSU_ERR_DEVICE_NEED_CALIBRATE_TOF          -212
00188 #define IBSU_ERR_DEVICE_INVALID_CALIBRATION_DATA    -213
00190 #define IBSU_ERR_DEVICE_HIGHER_SDK_REQUIRED         -214
00192 #define IBSU_ERR_DEVICE_LOCK_INVALID_BUFF           -215
00194 #define IBSU_ERR_DEVICE_LOCK_INFO_EMPTY             -216
```

```
00196 #define IBSU_ERR_DEVICE_LOCK_INFO_NOT_MATCHED      -217
00198 #define IBSU_ERR_DEVICE_LOCK_INVALID_CHECKSUM      -218
00200 #define IBSU_ERR_DEVICE_LOCK_INVALID_KEY           -219
00202 #define IBSU_ERR_DEVICE_LOCK_LOCKED                -220
00204 #define IBSU_ERR_DEVICE_LOCK_ILLEGAL_DEVICE        -221
00206 #define IBSU_ERR_DEVICE_LOCK_INVALID_SERIAL_FORMAT -222
00221 #define IBSU_ERR_CAPTURE_COMMAND_FAILED            -300
00223 #define IBSU_ERR_CAPTURE_STOP                      -301
00225 #define IBSU_ERR_CAPTURE_TIMEOUT                   -302
00227 #define IBSU_ERR_CAPTURE_STILL_RUNNING             -303
00229 #define IBSU_ERR_CAPTURE_NOT_RUNNING               -304
00231 #define IBSU_ERR_CAPTURE_INVALID_MODE              -305
00233 #define IBSU_ERR_CAPTURE_ALGORITHM                 -306
00235 #define IBSU_ERR_CAPTURE_ROLLING                   -307
00237 #define IBSU_ERR_CAPTURE_ROLLING_TIMEOUT           -308
00252 #define IBSU_ERR_CLIENT_WINDOW                     -400
00254 #define IBSU_ERR_CLIENT_WINDOW_NOT_CREATE          -401
00256 #define IBSU_ERR_INVALID_OVERLAY_HANDLE            -402
00271 #define IBSU_ERR_NBIS_NFIQ_FAILED                  -500
00273 #define IBSU_ERR_NBIS_WSQ_ENCODE_FAILED            -501
00275 #define IBSU_ERR_NBIS_WSQ_DECODE_FAILED            -502
00277 #define IBSU_ERR_NBIS_PNG_ENCODE_FAILED            -503
00279 #define IBSU_ERR_NBIS_JP2_ENCODE_FAILED            -504
00295 #define IBSU_ERR_DUPLICATE_EXTRACTION_FAILED        -600
00297 #define IBSU_ERR_DUPLICATE_ALREADY_USED             -601
00299 #define IBSU_ERR_DUPLICATE_SEGMENTATION_FAILED      -602
00301 #define IBSU_ERR_DUPLICATE_MATCHING_FAILED          -603
00316 #define IBSU_ERR_PAD_PROPERTY_DISABLED              -700
00331 #define IBSU_ERR_INCORRECT_STANDARD_FORMAT          -800
00347 #define IBSU_WRN_CHANNEL_IO_FRAME_MISSING           100
00349 #define IBSU_WRN_CHANNEL_IO_CAMERA_WRONG            101
00351 #define IBSU_WRN_CHANNEL_IO_SLEEP_STATUS            102
00353 #define IBSU_WRN_OUTDATED_FIRMWARE                  200
00355 #define IBSU_WRN_ALREADY_INITIALIZED                201
00357 #define IBSU_WRN_API_DEPRECATED                     202
00359 #define IBSU_WRN_ALREADY_ENHANCED_IMAGE             203
00361 #define IBSU_WRN_BGET_IMAGE                         300
00363 #define IBSU_WRN_ROLLING_NOT_RUNNING                301
00365 #define IBSU_WRN_NO_FINGER                          302
00367 #define IBSU_WRN_INCORRECT_FINGERS                  303
00369 #define IBSU_WRN_EMPTY_IBSM_RESULT_IMAGE            400
00371 #define IBSU_WRN_INVALID_BRIGHTNESS_FINGERS         600
00373 #define IBSU_WRN_WET_FINGERS                        601
00375 #define IBSU_WRN_MULTIPLE_FINGERS_DURING_ROLL       602
00377 #define IBSU_WRN_SPOOF_DETECTED                     603
00379 #define IBSU_WRN_ROLLING_SLIP_DETECTED              604
00381 #define IBSU_WRN_SPOOF_INIT_FAILED                  605
00382 /* No match from Matcher DB */
00383 #define IBSU_WRN_MATCHER_NO_MATCH                   700
00384 /* Finger is already registered */
00385 #define IBSU_WRN_MATCHER_ALREADY_REGISTERED         701
00386 /*
00387 ***************************************************************************************************
00388 * @}
00389 ***************************************************************************************************
00390 */
00391
00392
00401 #define IBSU_WRN_ROLLING_SMEAR                      304
00403 #define IBSU_WRN_ROLLING_SHIFTED_HORIZONTALLY       (IBSU_WRN_ROLLING_SMEAR | 1)
00405 #define IBSU_WRN_ROLLING_SHIFTED_VERTICALLY         (IBSU_WRN_ROLLING_SMEAR | 2)
00421 #define IBSU_WRN_QUALITY_INVALID_AREA               512
00423 #define IBSU_WRN_QUALITY_INVALID_AREA_HORIZONTALLY          (IBSU_WRN_QUALITY_INVALID_AREA | 1)
00425 #define IBSU_WRN_QUALITY_INVALID_AREA_VERTICALLY            (IBSU_WRN_QUALITY_INVALID_AREA | 2)
```

14.17 C_IncludeFiles/LinuxPort.h File Reference

```
#include <limits.h>
#include <inttypes.h>
#include <pthread.h>
```

Include dependency graph for LinuxPort.h:



Classes

- struct _GUID
- struct _SP_DEVICE_INTERFACE_DATA

Macros

- #define Sleep(x) usleep(x∗1000)
- #define AFX_MANAGE_STATE(x)
- #define CALLBACK
- #define WINAPI
- #define DECLSPEC_SELECTANY
- #define VOID void
- #define EXTERN_C extern "C"
- #define OVERLAPPED HANDLE
- #define MAX_PATH PATH_MAX
- #define DEFINE_GUID(name, l, w1, w2, b1, b2, b3, b4, b5, b6, b7, b8)
- #define INVALID_HANDLE_VALUE 0xFFFFFFFF
- #define TRUE 1
- #define FALSE 0
- #define RGB(r, g, b) ((COLORREF)(((BYTE)(r)|((WORD)((BYTE)(g))<<8))|(((DWORD)(BYTE)(b))<<16)))
- #define NULL 0

Typedefs

- typedef unsigned char UCHAR
- typedef unsigned char BYTE
- typedef unsigned short USHORT
- typedef unsigned int UINT
- typedef int LONG
- typedef unsigned int ULONG
- typedef UCHAR ∗ PUCHAR
- typedef unsigned int DWORD
- typedef unsigned short WORD
- typedef UCHAR TCHAR
- typedef DWORD HANDLE

- typedef int BOOL
- typedef char * LPSTR
- typedef char * LPCSTR
- typedef wchar_t * LPWSTR
- typedef wchar_t * LPCWSTR
- typedef wchar_t WCHAR
- typedef struct _GUID GUID
- typedef GUID * LPGUID
- typedef DWORD * LPDWORD
- typedef void * LPVOID
- typedef void * PVOID
- typedef void * LPCVOID
- typedef uint64_t ULONG_PTR
- typedef PVOID HDEVINFO
- typedef DWORD COLORREF
- typedef DWORD(WINAPI * PTHREAD_START_ROUTINE) (LPVOID lpThreadParameter)
- typedef PTHREAD_START_ROUTINE LPTHREAD_START_ROUTINE
- typedef int WPARAM
- typedef int LPARAM
- typedef HANDLE HWND
- typedef int LRESULT
- typedef struct _SP_DEVICE_INTERFACE_DATA SP_DEVICE_INTERFACE_DATA
- typedef struct _SP_DEVICE_INTERFACE_DATA * PSP_DEVICE_INTERFACE_DATA
- typedef PSP_DEVICE_INTERFACE_DATA PSP_INTERFACE_DEVICE_DATA

14.17.1 Macro Definition Documentation

14.17.1.1 AFX_MANAGE_STATE

```
#define AFX_MANAGE_STATE(
              x )
```

Definition at line 28 of file LinuxPort.h.

14.17.1.2 CALLBACK

```
#define CALLBACK
```

Definition at line 29 of file LinuxPort.h.

14.17.1.3 DECLSPEC_SELECTANY

```
#define DECLSPEC_SELECTANY
```

Definition at line 31 of file LinuxPort.h.

14.17.1.4 DEFINE_GUID

```
#define DEFINE_GUID(
              name,
              l,
              w1,
              w2,
              b1,
              b2,
              b3,
              b4,
              b5,
              b6,
              b7,
              b8 )
```

Value:

```
EXTERN_C const GUID DECLSPEC_SELECTANY name \
        = { l, w1, w2, { b1, b2,  b3,  b4,  b5,  b6,  b7,  b8 } }
```

Definition at line 97 of file LinuxPort.h.

14.17.1.5 EXTERN_C

```
#define EXTERN_C extern "C"
```

Definition at line 61 of file LinuxPort.h.

14.17.1.6 FALSE

```
#define FALSE 0
```

Definition at line 103 of file LinuxPort.h.

14.17.1.7 INVALID_HANDLE_VALUE

```
#define INVALID_HANDLE_VALUE 0xFFFFFFFF
```

Definition at line 101 of file LinuxPort.h.

14.17.1.8 MAX_PATH

```
#define MAX_PATH PATH_MAX
```

Definition at line 95 of file LinuxPort.h.

14.17.1.9 NULL

```
#define NULL 0
```

Definition at line 106 of file LinuxPort.h.

14.17.1.10 OVERLAPPED

```
#define OVERLAPPED HANDLE
```

Definition at line 73 of file LinuxPort.h.

14.17.1.11 RGB

```
#define RGB(
              r,
              g,
              b ) ((COLORREF)(((BYTE)(r)|((WORD)((BYTE)(g))<<8))|(((DWORD)(BYTE)(b))<<16)))
```

Definition at line 104 of file LinuxPort.h.

14.17.1.12 Sleep

```
#define Sleep(
              x ) usleep(x*1000)
```

Definition at line 27 of file LinuxPort.h.

14.17.1.13 TRUE

```
#define TRUE 1
```

Definition at line 102 of file LinuxPort.h.

14.17.1.14 VOID

```
#define VOID void
```

Definition at line 52 of file LinuxPort.h.

14.17.1.15 WINAPI

```
#define WINAPI
```

Definition at line 30 of file LinuxPort.h.

14.17.2 Typedef Documentation

14.17.2.1 BOOL

```
typedef int BOOL
```

Definition at line 44 of file LinuxPort.h.

14.17.2.2 BYTE

```
typedef unsigned char BYTE
```

Definition at line 34 of file LinuxPort.h.

14.17.2.3 COLORREF

```
typedef DWORD COLORREF
```

Definition at line 70 of file LinuxPort.h.

14.17.2.4 DWORD

```
typedef unsigned int DWORD
```

Definition at line 40 of file LinuxPort.h.

14.17.2.5 GUID

```
typedef struct _GUID GUID
```

14.17.2.6 HANDLE

```
typedef DWORD HANDLE
```

Definition at line 43 of file LinuxPort.h.

14.17.2.7 HDEVINFO

```
typedef PVOID HDEVINFO
```

Definition at line 69 of file LinuxPort.h.

14.17.2.8 HWND

```
typedef HANDLE HWND
```

Definition at line 78 of file LinuxPort.h.

14.17.2.9 LONG

```
typedef int LONG
```

Definition at line 37 of file LinuxPort.h.

14.17.2.10 LPARAM

```
typedef int LPARAM
```

Definition at line 77 of file LinuxPort.h.

14.17.2.11 LPCSTR

```
typedef char* LPCSTR
```

Definition at line 47 of file LinuxPort.h.

14.17.2.12 LPCVOID

```
typedef void* LPCVOID
```

Definition at line 66 of file LinuxPort.h.

14.17.2.13 LPCWSTR

```
typedef wchar_t* LPCWSTR
```

Definition at line 49 of file LinuxPort.h.

14.17.2.14 LPDWORD

```
typedef DWORD* LPDWORD
```

Definition at line 63 of file LinuxPort.h.

14.17.2.15 LPGUID

```
typedef GUID* LPGUID
```

Definition at line 62 of file LinuxPort.h.

14.17.2.16 LPSTR

```
typedef char* LPSTR
```

Definition at line 46 of file LinuxPort.h.

14.17.2.17 LPTHREAD_START_ROUTINE

```
typedef PTHREAD_START_ROUTINE LPTHREAD_START_ROUTINE
```

Definition at line 75 of file LinuxPort.h.

14.17.2.18 LPVOID

```
typedef void* LPVOID
```

Definition at line 64 of file LinuxPort.h.

14.17.2.19 LPWSTR

```
typedef wchar_t* LPWSTR
```

Definition at line 48 of file LinuxPort.h.

14.17.2.20 LRESULT

```
typedef int LRESULT
```

Definition at line 79 of file LinuxPort.h.

14.17.2.21 PSP_DEVICE_INTERFACE_DATA

```
typedef struct _SP_DEVICE_INTERFACE_DATA * PSP_DEVICE_INTERFACE_DATA
```

14.17.2.22 PSP_INTERFACE_DEVICE_DATA

```
typedef PSP_DEVICE_INTERFACE_DATA PSP_INTERFACE_DEVICE_DATA
```

Definition at line 93 of file LinuxPort.h.

14.17.2.23 PTHREAD_START_ROUTINE

```
typedef DWORD(WINAPI * PTHREAD_START_ROUTINE) (LPVOID lpThreadParameter)
```

Definition at line 74 of file LinuxPort.h.

14.17.2.24 PUCHAR

```
typedef UCHAR* PUCHAR
```

Definition at line 39 of file LinuxPort.h.

14.17.2.25 PVOID

```
typedef void* PVOID
```

Definition at line 65 of file LinuxPort.h.

14.17.2.26 SP_DEVICE_INTERFACE_DATA

```
typedef struct _SP_DEVICE_INTERFACE_DATA SP_DEVICE_INTERFACE_DATA
```

14.17.2.27 TCHAR

```
typedef UCHAR TCHAR
```

Definition at line 42 of file LinuxPort.h.

14.17.2.28 UCHAR

```
typedef unsigned char UCHAR
```

Definition at line 33 of file LinuxPort.h.

14.17.2.29 UINT

```
typedef unsigned int UINT
```

Definition at line 36 of file LinuxPort.h.

14.17.2.30 ULONG

```
typedef unsigned int ULONG
```

Definition at line 38 of file LinuxPort.h.

14.17.2.31 ULONG_PTR

```
typedef uint64_t ULONG_PTR
```

Definition at line 68 of file LinuxPort.h.

14.17.2.32 USHORT

```
typedef unsigned short USHORT
```

Definition at line 35 of file LinuxPort.h.

14.17.2.33 WCHAR

```
typedef wchar_t WCHAR
```

Definition at line 50 of file LinuxPort.h.

14.17.2.34 WORD

```
typedef unsigned short WORD
```

Definition at line 41 of file LinuxPort.h.

14.17.2.35 WPARAM

```
typedef int WPARAM
```

Definition at line 76 of file LinuxPort.h.

14.18 LinuxPort.h

Go to the documentation of this file.

```
00001 /*
00002 ********************************************************************************************************
00003 * LinuxPort.h
00004 *
00005 * DESCRIPTION:
00006 *     Types and constants for IBScanUltimate Linux port.
00007 *     http://www.integratedbiometrics.com
00008 *
00009 * NOTES:
00010 *     Copyright (c) Integrated Biometrics, 2009-2013
00011 *
00012 * HISTORY:
00013 *     2012/04/06  1.0.0  Created.
00014 *     2013/08/03  1.6.9  Reformatted.
00015 *     2014/07/23  1.8.0  Removed typedef (struct _GUID)
00016 *     2015/03/04  1.8.3  Added typedef keyword to support UNICODE for WinCE
00017 ********************************************************************************************************
00018 */
00019
00020 #ifndef __LINUX_PORT_H__
00021 #define __LINUX_PORT_H__
00022
00023 #include <limits.h>
00024 #include <inttypes.h>
00025 #include <pthread.h>
00026
00027 #define Sleep(x) usleep(x*1000)
00028 #define AFX_MANAGE_STATE(x)
00029 #define CALLBACK
00030 #define WINAPI
00031 #define DECLSPEC_SELECTANY
00032
00033 typedef unsigned char   UCHAR;
00034 typedef unsigned char   BYTE;
00035 typedef unsigned short  USHORT;
00036 typedef unsigned int    UINT;
00037 typedef int             LONG;
00038 typedef unsigned int    ULONG;
00039 typedef UCHAR          *PUCHAR;
00040 typedef unsigned int    DWORD;
00041 typedef unsigned short  WORD;
00042 typedef UCHAR           TCHAR;
00043 typedef DWORD           HANDLE;
00044 typedef int             BOOL;
00045
00046 typedef char           *LPSTR;
00047 typedef char           *LPCSTR;
00048 typedef wchar_t        *LPWSTR;
00049 typedef wchar_t        *LPCWSTR;
00050 typedef wchar_t         WCHAR;
00051
00052 #define VOID void
00053
00054 typedef struct _GUID {
00055     DWORD  Data1;
00056     USHORT Data2;
00057     USHORT Data3;
00058     BYTE   Data4[ 8 ];
00059 } GUID;
00060
00061 #define EXTERN_C extern "C"
00062 typedef GUID                *LPGUID;
00063 typedef DWORD               *LPDWORD;
00064 typedef void                *LPVOID;
00065 typedef void                *PVOID;
00066 typedef void                *LPCVOID;
00067
00068 typedef uint64_t             ULONG_PTR;
00069 typedef PVOID                HDEVINFO;
00070 typedef DWORD                COLORREF;
00071
00072 // Fixup type later
00073 #define OVERLAPPED HANDLE
00074 typedef DWORD        (WINAPI *PTHREAD_START_ROUTINE)(LPVOID lpThreadParameter);
00075 typedef PTHREAD_START_ROUTINE LPTHREAD_START_ROUTINE;
00076 typedef int                  WPARAM;
00077 typedef int                  LPARAM;
00078 typedef HANDLE               HWND;
00079 typedef int                  LRESULT;
00080
00081 //
00082 // Device interface information structure (references a device
```

```
00083 // interface that is associated with the device information
00084 // element that owns it).
00085 //
00086 typedef struct _SP_DEVICE_INTERFACE_DATA {
00087     DWORD cbSize;
00088     GUID  InterfaceClassGuid;
00089     DWORD Flags;
00090     ULONG_PTR Reserved;
00091 } SP_DEVICE_INTERFACE_DATA, *PSP_DEVICE_INTERFACE_DATA;
00092
00093 typedef PSP_DEVICE_INTERFACE_DATA PSP_INTERFACE_DEVICE_DATA;
00094
00095 #define MAX_PATH PATH_MAX
00096
00097 #define DEFINE_GUID(name, l, w1, w2, b1, b2, b3, b4, b5, b6, b7, b8) \
00098         EXTERN_C const GUID DECLSPEC_SELECTANY name \
00099                 = { l, w1, w2, { b1, b2,  b3,  b4,  b5,  b6,  b7,  b8 } }
00100
00101 #define INVALID_HANDLE_VALUE 0xFFFFFFFF
00102 #define TRUE  1
00103 #define FALSE 0
00104 #define RGB(r,g,b)          ((COLORREF)(((BYTE)(r)|((WORD)((BYTE)(g))«8))|(((DWORD)(BYTE)(b))«16)))
00105 #ifndef NULL
00106 #define NULL  0
00107 #endif
00108
00109
00110 #endif  // __LINUX_PORT_H__
```

14.19 C_IncludeFiles/mainpage.dox File Reference

14.20 C_IncludeFiles/overview.dox File Reference

14.21 C_IncludeFiles/revision history.dox File Reference

14.22 C_IncludeFiles/structures and Constants.dox File Reference

14.23 C_IncludeFiles/support.dox File Reference

Index