IBScanUltimate API Manual for Java (and Android)

Version 3.2.0(Jan 22, 2020)

Table of Contents

Revision History

Date	Author	Remarks
2020/01	Milton	Added descriptions of new PropertyID values for IBScanUltimate v3.2.0 : IS_SPOOF_SUPPORTED, ENABLE_SPOOF, SPOOF_LEVEL
2019/06	Milton	Added SetEncryption() method on Android Added description of new IBScanDevice nested enumeration EncyptionMode
2018/11	Milton	Added description of new IBScan method getRequiredSDKVersion() Added SavePngImage() method on Android Added SaveJP2Image() method on Android Added SaveBitmapImage() method on Android
2018/4	YOUNG	Added description of new IBScanDevice method generateDisplayImage() Added description of new IBScanDevice method removeFingerImage() Added description of new IBScanDevice method addFingerImage() Added description of new IBScanDevice method isFingerDuplicated() Added description of new IBScanDevice method isValidFingerGeometry() Added description of new IBScanDevice method SaveBitmapImage()
2017/4	Gon	Added description of new IBScanDevice method getCombineImageEx(). Added descriptions of new PropertyID values for IBScanUltimate v1.9.6:. NO_PREVIEW_IMAGE, ROLL_IMAGE_OVERRIDE, WARNING_MESSAGE_INVALID_AREA, ENABLE_WET_FINGER_DETECT, WET_FINGER_DETECT_LEVE. Added descriptions of new FingerQualityState values for IBScanUltimate v1.9.6: INVALID_AREA_BOTTOM. Added descriptions of new warning code for IBScanUltimate v1.9.6: QUALITY_INVALID_AREA, QUALITY_INVALID_AREA_HORIZONTALLY, QUALITY_INVALID_AREA_VERTICALLY, QUALITY_INVALID_AREA_HORIZONTALLY_VERTICALLY, INVALID_BRIGHTNESS_FINGERS, WET_FINGERS
2016/4	Yesica	Added descriptions of new PropertyID values for IBScanUltimate v1.9.3: ROLLED_IMAGE_WIDTH and ROLLED_IMAGE_HEIGHT.
2015/12	Yesica	Added description of new IBScanDevice method getOperableBeeper (). Added description of new IBScanDevice method setBeeper (). Added description of new IBScanDeviceListener method devicePressedKeyButtons ().
2015/8	Yesica	Added description of new IBScanDevice method getCombineImage (). Added description of new IBScanDevice nested enumeration

		CombineImageWhichHand.
2013/10	BAN	Added descriptions of new PropertyID values for IBScanUltimate v1.7.0: CAPTURE_TIMEOUT and ROLL_MIN_WIDTH. Added descriptions of new FingerQualityState values for IBScanUltimate v1.7.0: INVALID_AREA_TOP, INVALID_AREA_LEFT, INVALID_AREA_RIGHT. Added description of new IBScanDevice method captureImageExtended(). Added description of new IBScanDevice nested class SegmentPosition. Added description of new IBScanDeviceListener method deviceImageResultExtendedAvailable(). Added description of new IBScanDevice method enableEvent(). Added description of new IBScanDevice nested enumeration EventType.

1. API Descriptions

1.1. Java API Summary

The Java API of IBScanUltimate is contained within the package com.integratedbiometrics.ibscanultimate. Addition common definitions are contained within the package com.integratedbiometrics.ibscancommon.

Some methods, marked with asterisks, are available only on Android.

1.1.1. Summary of Main Class and Interface Methods

No	Method
IBScan methods (all platforms)	
1	public void **enableLogTrace**(boolean on) throws IBScanException
2	public int **getDeviceCount**() throws IBScanException
3	public IBScan.DeviceDesc **getDeviceDescription**(int deviceIndex) throws IBScanException
4	public int **getInitProgress**(int deviceIndex) throws IBScanException
5	public static IBScan **getInstance**()
6*	public static IBScan **getInstance**(Context context)
7	public IBScan.SdkVersion **getSdkVersion**() throws IBScanException
8*	public boolean **hasPermission**(int deviceId)
9*	public static boolean **isScanDevice**(UsbDevice device)
10	public IBScanDevice **openDevice**(int deviceIndex) throws IBScanException
11	public IBScanDevice **openDevice**(int deviceIndex,String uniformityMaskPath) throws IBScanException
12	public void **openDeviceAsync**(int deviceIndex) throws IBScanException
13	public void **openDeviceAsync**(int deviceIndex, String uniformityMaskPath) throws IBScanException
14*	public void **requestPermission**(int deviceId)
15*	public void **setContext**(Context context)
16	public void **setScanListener**(IBScanListener listener)
17	public String **getRequiredSDKVersion**(final int deviceIndex) throws IBScanException
IBScanDevice methods	
1	public void **beginCaptureImage**(IBScanDevice.ImageType imageType, IBScanDevice.ImageResolution imageResolution, int captureOptions) throws IBScanException
2	public int **calculateNfiqScore**(ImageData image) throws IBScanException
3	public void **cancelCaptureImage**() throws IBScanException
4	public java.lang.Object[] **captureImage**() throws IBScanException
5	public java.lang.Object[] **captureImageExtended**() throws IBScanException

6	public void **captureImageManually**() throws IBScanException
7	public void **close**() throws IBScanException
8	public void **enableEvent**(EventType event, boolean enable) throws IBScanException
9	public int **getContrast**() throws IBScanException
10	public long **getLEDs**() throws IBScanException
11	public IBScanDevice.LEOperationMode **getLEOperationMode**() throws IBScanException
12	public IBScanDevice.LedState **getOperableLEDs**() throws IBScanException
13	public IBScanDevice.PlatenState **getPlatenStateAtCapture**() throws IBScanException
14	Public String **getProperty**(IBScanDevice.PropertyId propertyId) throws IBScanException
15	public Object[] **getResultImageInfo**(IBCommon.FingerPosition fingerPosition) throws IBScanException
16	public IBScanDevice.RollingData **getRollingInfo**() throws IBScanException
17	public boolean **isCaptureActive**() throws IBScanException
18	public boolean **isCaptureAvailable**(IBScanDevice.ImageType imageType, IBScanDevice.ImageResolution imageResolution) throws IBScanException
19	public boolean **isFingerTouching**() throws IBScanException
20	public boolean **isOpened**()
21	public void **setContrast**(int contrastValue) throws IBScanException
22	public void **setLEDs**(long activeLEDs) throws IBScanException
23	public void **setLEOperationMode**(IBScanDevice.LEOperationMode leOperationMode) throws IBScanException
24	public void **setProperty**(IBScanDevice.PropertyId propertyId, String propertyValue) throws IBScanException
25	public void **setScanDeviceListener**(IBScanDeviceListener listener)
26	public Object **getCombineImage**(IBScanDevice.ImageData inImage1, IBScanDevice.ImageData inImage2,int whichHand)
27	Public int **getOperableBeeper**(BeeperType bType)
28	Public void **setBeeper(**BeepPattern bPattern,long soundTone,long duration,long reserved_1,long reserved_2)
29	public Object[] **getCombineImageEx**(IBScanDevice.ImageData inImage1, IBScanDevice.ImageData inImage2,int whichHand)
30	public int **generateDisplayImage**(byte[] image, int inWidth, int inHeight, byte[]outImage, int outWidth, int outHeight, byte bkColor, int outFormat, int outQualityLevel, Boolean outVerticalFlip)
31	public int **removeFingerImage**(long fIndex)
32	public int **addFingerImage**(IBScanDevice.ImageData image, long fIndex, IBScanDevice.ImageType imageType, boolean flagForce)
33	public long **isFingerDuplicated**(IBScanDevice.ImageData image, long fIndex,

	IBScanDevice.ImageType imageType, int securityLevel)
34	public boolean **isValidFingerGeometry**(IBScanDevice.ImageData image, long fIndex, IBScanDevice.ImageType imageType)
IBScanListener methods	
1*	void **scanDeviceAttached**(int deviceId)
2*	void **scanDeviceDetached**(int deviceId)
3	void **scanDeviceCountChanged**(int deviceCount)
4	void **scanDeviceInitProgress**(int deviceIndex, int progressValue)
5	void **scanDeviceOpenComplete**(int deviceIndex, IBScanDevice device, IBScanException exception)
6*	void **scanDevicePermissionGranted**(int deviceId, boolean granted)
IBScanDeviceListener methods	
1	void **deviceCommunicationBroken**(IBScanDevice device)
2	void **deviceImagePreviewAvailable**(IBScanDevice device, IBScanDevice.ImageData image)
3	void **deviceFingerCountChanged**(IBScanDevice device, IBScanDevice.FingerCountState fingerState)
4	void **deviceFingerQualityChanged**(IBScanDevice device, IBScanDevice.FingerQualityState[] fingerQualities)
5	void **deviceAcquisitionBegun**(IBScanDevice device, IBScanDevice.ImageType imageType)
6	void **deviceAcquisitionCompleted**(IBScanDevice device, IBScanDevice.ImageType imageType)
7	void **deviceImageResultAvailable**(IBScanDevice device, IBScanDevice.ImageData image, IBScanDevice.ImageType imageType, IBScanDevice.ImageData[] splitImageArray)
8	void **deviceImageResultExtendedAvailable**(IBScanDevice device, IBScanException imageStatus, IBScanDevice.ImageData image, IBScanDevice.ImageType imageType, int detectedFingerCount, IBScanDevice.ImageData[] segmentImageArray, SegmentPosition[] segmentPositionArray)
9	void **devicePlatenStateChanged**(IBScanDevice device, IBScanDevice.PlatenState platenState)
10	void **deviceWarningReceived**(IBScanDevice device, IBScanException warning)
11	void **devicePressedKeyButtons**(IBScanDevice device,int pressedKeyButtons)

Table 1
*** Available only on Android**

1.1.2. Summary of Nested Classes

No	Class
IBScan nested classes	
1	public static class IBScan.**DeviceDesc**

2	public static class IBScan.**SdkVersion**
IBScanDevice nested classes	
1	public static enum IBScanDevice.**EventType**
2	public static enum IBScanDevice.**FingerCountState**
3	public static enum IBScanDevice.**FingerQualityState**
4	public static class IBScanDevice.**ImageData**
5	public static enum IBScanDevice.**ImageFormat**
6	public static enum IBScanDevice.**ImageResolution**
7	public static enum IBScanDevice.**ImageType**
8	public static class IBScanDevice.**LedState**
9	public static enum IBScanDevice.**LedType**
10	public static enum IBScanDevice.**LEOperationMode**
11	public static enum IBScanDevice.**PlatenState**
12	public static enum IBScanDevice.**PropertyId**
13	public static class IBScanDevice.**RollingData**
14	public static enum IBScanDevice.**RollingState**
15	public static class IBScanDevice.**SegmentPosition**

Table 2

1.1.3. Summary of Common Classes

No	Class
IBCommon nested classes	
1	public static class IBCommon.**ImageDataExt**
2	public static enum IBCommon.**ImageFormat**
3	public static enum IBCommon.**ImpressionType**
4	public static enum IBCommon.**FingerPosition**
5	public static enum IBCommon.**CaptureDeviceTechId**

Table 3

1.1.4. Class IBScan

1.1.4.1. Summary

The single instance of this class may be gotten with getInstance(). The application will typically register a IBScanListener to receive notifications for events such as device count change and device communication failure. Device instances should be obtained by either the blocking openDevice() method or non-blocking openDeviceAsync() method.

On Android, the Activity accessing IB scanners must set the context for operations with setContext(). Several more Android-specific functions are provided to manage USB devices.

1.1.4.2. enableTraceLog()

- ***Signature***

Method	public void **enableTraceLog**(boolean on) throws IBScanException

- ***Description***

Enables or disable trace log in native library. The trace log is enabled by default.

- ***Parameter***

Parameter	Description
on	true to enable trace log; false to disable trace log

1.1.4.3. getDeviceCount()

- ***Signature***

Method	public int **getDeviceCount**() throws IBScanException

- ***Description***

Retrieve count of connected IB USB scanner devices. Only the attached devices to which the caller has been granted permission will be counted.

- ***Return***

count of IB USB scanner devices

1.1.4.4. getDeviceDescription()

- ***Signature***

Method	public IBScan.DeviceDesc **getDeviceDescription**(int deviceIndex) throws IBScanException

- ***Description***

Retrieve detailed device information about particular scanner by logical index.

- ***Parameter***

Parameter	Description
deviceIndex	zero-based index of the device

- ***Return***

a description of the device

1.1.4.5. getInitProgress()

- ***Signature***

Method	public int **getInitProgress**(int deviceIndex) throws IBScanException

- ***Description***

Get initialization progress.

- ***Parameter***

Parameter	Description
deviceIndex	zero-based index of the device

- ***Return***

initialization progress between 0 and 100. A value of 100 indicates that that initialization is complete

1.1.4.6. getInstance()

- ***Signature***

Method	public static IBScan **getInstance**()

- ***Description***

Get single instance of class. On Android, a context must be associated with the instance through a later call to setContext().

- ***Return***

single instance of IBScan

1.1.4.7. getInstance(Context) (Android only)

- ***Signature***

Method	public static IBScan **getInstance**(Context context)

- ***Description***

  Get single instance of class.

- ***Parameter***

Parameter	Description
context	the context for the receiver and USB accesses

- ***Return***

  single instance of IBScan

1.1.4.8. getSdkVersion()

- ***Signature***

Method	public IBScan.SdkVersion **getSdkVersion**() throws IBScanException

- ***Description***

  Obtains product and software version information.

- ***Return***

  SDK version

1.1.4.9. hasPermission() (Android only)

- ***Signature***

Method	public boolean **hasPermission**(int deviceId)

- ***Description***

  Determines whether the caller has permission to access the device.

- ***Parameter***

Parameter	Description
deviceId	ID of the device. This is the ID that Android assigns to the device, obtained through the UsbDevice getDeviceId() method.

- ***Return***

  true if caller has permission; false otherwise.

1.1.4.10. isScanDevice() (Android only)

- ***Signature***

Method	public static boolean **isScanDevice**(UsbDevice device)

- ***Description***

  Determine whether device is a scan device. This just checks whether the vendor and product IDs match recognized devices.

- ***Parameter***

Parameter	Description
device	device to investigate

- ***Return***

  true if device is an IB scan device; false otherwise

1.1.4.11. openDevice(int)

- ***Signature***

Method	public IBScanDevice **openDevice**(int deviceIndex) throws IBScanException

- ***Description***

  Initialize device, given a particular by device index. This function blocks until an error occurs or initialization completes; meanwhile any registered IBScanListener will receive scanDeviceInitProgress() callbacks to track the initialization progress. Either a device object will be returned to the application or an exception will be thrown.

- ***Parameter***

Parameter	Description
deviceIndex	zero-based index of the device

- ***Return***

  device object, if initialization succeeds; null otherwise

1.1.4.12. openDevice(int, String)

- ***Signature***

Method	public IBScanDevice **openDevice**(int deviceIndex, String uniformityMaskPath) throws IBScanException

- ***Description***

  See also openDevice(int).

- ***Parameter***

Parameter	Description

deviceIndex	zero-based index of the device
uniformityMaskPath	uniformity mask path

- ***Return***

  device object, if initialization succeeds; null otherwise

1.1.4.13. openDeviceAsync(int)

- ***Signature***

Method	public void **openDeviceAsync**(int deviceIndex) throws IBScanException

- ***Description***

  Initialize device asynchronously, given a particular by device index. This function returns immediately. Any registered IBScanListener will receive scanDeviceInitProgress() callbacks to track the initialization progress. When an error occurs or initialization completes, scanDeviceOpenComplete() will be invoked with either a device object or a description of the error that occurred.

- ***Parameter***

Parameter	Description
deviceIndex	zero-based index of the device

1.1.4.14. openDeviceAsync(int, String)

- ***Signature***

Method	public void **openDeviceAsync**(int deviceIndex, String uniformityMaskPath) throws IBScanException

- ***Description***

  See also openDeviceAsync(int).

- ***Parameter***

Parameter	Description
deviceIndex	zero-based index of the device
uniformityMaskPath	uniformity mask path

1.1.4.15. requestPermission() (Android only)

- ***Signature***

Method	public void **requestPermission**(int deviceId)

- ***Description***

  Request permission to access the device. Success or failure will be returned to the user through the registered IBScanListener's scanDevicePermissionGranted() callback. If permission has not already been granted to the device, a dialog may be shown to the user.

- ***Parameter***

Parameter	Description
deviceId	ID of the device. This is the ID that Android assigns to the device, obtained through the UsbDevice getDeviceId() method.

1.1.4.16. setContext() (Android only)

- ***Signature***

Method	public void **setContext**(Context context)

- ***Description***

  Set the context for this IBScan. This function must be called by an activity for scanner devices to be recognized and accessible. When one activity transfers control of the IB scanners to another, this function should be called with a null argument to release the reference to the context and unregister it as a USB receiver.

- ***Parameter***

Parameter	Description
context	the context for the reciever and USB accesses. If null, the existing context will still be unregistered as a receiver and the reference to it will be cleared.

1.1.4.17. setScanListener()

- ***Signature***

Method	public void **setScanListener**(IBScanListener listener)

- ***Description***

  Set listener for scan events.

- ***Parameter***

Parameter	Description
listener	listener for scan events

1.1.4.18. unloadLibrary()

- ***Signature***

Method	public void **unloadLibrary**()

- ***Description***

  Release the library from the address space manually.

1.1.4.19. updateUsbPermission()(Android only)

- ***Signature***

Method	public void **updateUsbPermission**()

- ***Description***

  Update the permission to allow the approach to attached USB bus by libusb library. It is required rooting of Android device.

1.1.4.20. getRequiredSDKVersion()

- ***Signature***

Method	public String **getRequiredSDKVersion**(int deviceIndex)

- ***Description***

  Get minimum SDK version required for running.

1.1.5. Class IBScanDevice

1.1.5.1. Summary

Principal class for interfacing with particular IB scanners.

1.1.5.2. beginCaptureImage()

- ***Signature***

Method	public void **beginCaptureImage**(IBScanDevice.ImageType imageType, IBScanDevice.ImageResolution imageResolution, int captureOptions) throws IBScanException

- ***Description***

Start image acquisition for the device. This function will return immediately, but this device's IBScanDeviceListener will inform the application about scanning progress with the methods deviceFingerCountChanged(), deviceFingerQualityChanged, devicePlatenStateChanged(), and deviceImagePreviewAvailable(). When a quality scan with the correct number of fingers is available or captureImageManually() prematurely aborts the scan, the listener's deviceImageResultAvailable() method will supply a final scan to the application.

- ***Parameter***

Parameter	Description
imageType	the image type of the image to acquire
imageResolution	the image resolution of the image to acquire
captureOptions	a bit-mapped value indicating capture options, consisting of zero or more options OR'd together • OPTION_AUTO_CAPTURE auto capture • OPTION_AUTO_CONTRAST auto contrast • OPTION_IGNORE_FINGER_COUNT ignore finger count

1.1.5.3. calculateNfiqScore()

- ***Signature***

Method	public int **calculateNfiqScore**(ImageData image) throws IBScanException

- ***Description***

Calculate NFIQ score for image. The calculation is potentially long-running and

may should be performed on a background thread.

- ***Parameter***

Parameter	Description
image	the image for which the NFIQ score will be calculate

- ***Return***

The NFIQ score, between 1 (best) and 5 (worst), inclusive.

1.1.5.4. cancelCaptureImage()

- ***Signature***

Method	public void **cancelCaptureImage**() throws IBScanException

- ***Description***

Abort image acquisition on the device. After beginCaptureImage() is called, image capture can be prematurely terminated with this function..

1.1.5.5. captureImage()

- ***Signature***

Method	public Object[] **captureImage**() throws IBScanException

- ***Description***

Capture an image from scanner.

- ***Return***

an array containing information about captured image. The contents of the returned array, in order, are

- ImageData image - image data of preview image or result image
- ImageType imageType - image type
- ImageData[] splitImageArray - finger array split from the result image
- FingerCountState fingerCountState - finger count state
- FingerQualityState[] qualityArray - finger quality states

1.1.5.6. captureImageExtended()

- ***Signature***

Method	public Object[] **captureImageExtended**() throws IBScanException

- ***Description***

Capture an image from scanner, returning extended information.

- ***Return***

an array containing information about captured image. The contents of the returned array, in order, are

- IBScanException image status – status from result image acquisition
- ImageData image - image data of preview image or result image
- ImageType imageType - image type
- Integer detectedFIngerCount – detected finger count
- ImageData[] segmentImageArray - finger array split from the result image
- SegmentPosition[] segmentPositionArray – position data for individual fingers split from result image
- FingerCountState fingerCountState - finger count state
- FingerQualityState[] qualityArray - finger quality states

1.1.5.7. captureImageManually()

- ***Signature***

Method	public void **captureImageManually**() throws IBScanException

- ***Description***

Capture current scanner image as result image. After beginCaptureImage() is called, scanning typically continues until a quality scan with the correct number of fingers is available or an error occurs. This function will prematurely terminate the process and return the current scanner image to the application with the IBScanDeviceListener's deviceImageResultAvailable() method.

1.1.5.8. close()

- ***Signature***

Method	public void close() throws IBScanException

- ***Description***

Release a device.

1.1.5.9. enableEvent()

- ***Signature***

Method	public void **enableEvent**(EventType event, boolean enable) throws IBScanException

- ***Description***

Enable or disable a low-level event for this device. When a device is opened all events are enabled. Disabling an event will effectively disable the associated method in the configured IBScanDeviceListener.

- ***Parameter***

Parameter	Description
event	Event to enable or disable
enable	true to enable event; false to disable event

1.1.5.10. getContrast()

- ***Signature***

Method	public int **getContrast**() throws IBScanException

- ***Description***

Get the contrast value for the device.

- ***Return***

contrast value between MIN_CONTRAST_VALUE and MAX_CONTRAST_VALUE, inclusive

1.1.5.11. getLEDs()

- ***Signature***

Method	public long **getLEDs**() throws IBScanException

- ***Description***

Get the active status LEDs of the device.

- ***Return***

the bit-mapped status of the LEDs; set bits indicate "on" LEDs

1.1.5.12. getLEOperationMode()

- ***Signature***

Method	public IBScanDevice.LEOperationMode **getLEOperationMode**() throws IBScanException

- ***Description***

Get the Light-Emitting (LE) film operation mode (On, Off, or Auto) for the device.

- ***Return***

light-emitting film operation mode

1.1.5.13. getOperableLEDs()

- ***Signature***

Method	public IBScanDevice.LedState **getOperableLEDs**() throws IBScanException

- ***Description***

Get a description of the operable status LEDs of the device.

- ***Return***

a description of the status LEDs

1.1.5.14. getPlatenStateAtCapture()

- ***Signature***

Method	public IBScanDevice.PlatenState **getPlatenStateAtCapture**() throws IBScanException

- ***Description***

Get information about platen state when capture was started.

- ***Return***

information about platen state

1.1.5.15. getProperty()

- ***Signature***

Method	public String **getProperty**(IBScanDevice.PropertyId propertyId) throws IBScanException

- ***Description***

Retrieves a property value from the device.

- ***Parameter***

Parameter	Description
propertyId	the ID of the property to get

- ***Return***

the value of the property, as a string

1.1.5.16. getResultImageExt()

- ***Signature***

Method	public Object[] **getResultImageExt**(IBCommon.FingerPosition fingerPosition) throws IBScanException

- ***Description***

Get extended result image information.

- ***Parameter***

Parameter	Description
fingerPosition	finger position of finger(s) captured

- ***Return***

an array containing information about captured image. The contents of the returned array, in order, are

- IBCommon.ImageDataExt image - image data of preview image or result image
- IBCommon.ImageDataExt[] splitImageArray - finger array split from the result image

1.1.5.17. getRollingInfo()

- ***Signature***

Method	public IBScanDevice.RollingData **getRollingInfo**() throws IBScanException

- ***Description***

Get information about rolling status.

- ***Return***

information about rolling status

1.1.5.18. isCaptureActive()

- ***Signature***

Method	public boolean **isCaptureActive**() throws IBScanException

- ***Description***

Check if capture is active on the device.

- ***Return***

true if capture is active; false otherwise

1.1.5.19. isCaptureAvailable()

- ***Signature***

Method	public boolean **isCaptureAvailable**(IBScanDevice.ImageType imageType, IBScanDevice.ImageResolution imageResolution) throws IBScanException

- ***Description***

Check if requested capture mode is supported by the device.

- ***Parameter***

Parameter	Description
imageType	the image type of the mode to check
imageResolution	the image resolution of the mode to check

- ***Return***

true if the mode is supported; false otherwise

1.1.5.20. isFingerTouching()

- ***Signature***

Method	public boolean **isFingerTouching**() throws IBScanException

- ***Description***

Determines if one or more fingers is currently touching the detector.

- ***Return***

true if a finger is on the detector; false otherwise

1.1.5.21. isOpened()

- ***Signature***

Method	public boolean **isOpened**()

- ***Description***

Check if a particular device is open/initialized.

- ***Return***

true if the device is open; false otherwise

1.1.5.22. setContrast()

- ***Signature***

Method	public void **setContrast**(int contrastValue) throws IBScanException

- ***Description***

Set the contrast value for the device.

- ***Parameter***

Parameter	Description
contrastValue	contrast value between MIN_CONTRAST_VALUE and MAX_CONTRAST_VALUE, inclusive, to set

1.1.5.23. setLEDs()

- ***Signature***

Method	public void **setLEDs**(long activeLEDs) throws IBScanException

- ***Description***

Set the active status of LEDs of the device.

- ***Parameter***

Parameter	Description
activeLEDs	the bit-mapped status of the LEDs; set bits indicate LEDs to turn on, clear bits indicate LEDs to turn off

1.1.5.24. setLEOperationMode()

- ***Signature***

Method	public void **setLEOperationMode**(IBScanDevice.LEOperationMode leOperationMode) throws IBScanException

- ***Description***

Set the Light-Emitting (LE) film operation mode (On, Off, or Auto) the device.

- ***Parameter***

Parameter	Description
leOperationMode	light-emitting film operation mode to set

1.1.5.25. setProperty()

- ***Signature***

Method	public void **setProperty**(IBScanDevice.PropertyId propertyId, String propertyValue) throws IBScanException

- ***Description***

Set a property value of the device.

- ***Parameter***

Parameter	Description
propertyId	the ID of the property to set
propertyValue	the value to set for the property, as a string

1.1.5.26. setPropertyReserved()

- ***Signature***

Method	public void **setPropertyReserved**(String reservedKey, IBScanDevice.PropertyId propertyId, String propertyValue) throws IBScanException

- ***Description***

  Set a reserved property value of the device. It need to get reserved key from Integrated Biometrics to use this method.

- ***Parameter***

Parameter	Description
reservedKey	The reserved key to set for the property, as a string
propertyId	the ID of the property to set
propertyValue	the value to set for the property, as a string

1.1.5.27. setScanDeviceListener()

- ***Signature***

Method	public void **setScanDeviceListener**(IBScanDeviceListener listener)

- ***Description***

Register listener for scan device events.

- ***Parameter***

Parameter	Description
listener	listener for events

1.1.5.28. generateZoomOutImageEx()

- ***Signature***

Method	public int **generateZoomOutImageEx**(byte[] image, int inWidth, int inHeight, byte[]outImage, int outWidth, int outHeight, byte bkColor) throws IBScanException

- ***Description***

  Generate scaled version of image.

- ***Parameter***

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

1.1.5.29. wsqEncodeToFile()

- ***Signature***

Method	public int **wsqEncodeToFile**(String filename, byte[] image, int width, int height, int pitch, int bitPerPixel, int pixelPerInch, double bitrate, String commentText) throws IBScanException

- ***Description***

Save WSQ compresses grayscale fingerprint image to specific file path.

- ***Parameter***

Parameter	Description
filename	File path to save image which is compressed from original image by WSQ compression
image	Original image data
width	Width of original image
height	Height of original image
pitch	Image line pitch (in bytes). A positive value indicates top-down line order; a negative value indicates bottom-up line order
bitPerPixel	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

1.1.5.30. SavePngImage()

- ***Signature***

Method	public int **SavePngImage**(String filename, byte[] image, int width, int height, int pitch, double resX, double resY) throws IBScanException

- ***Description***

  Save png image to specific file path.

- ***Parameter***

Parameter	Description
filename	File path to save png image
image	Original image data
width	Width of original image
height	Height of original image
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)

1.1.5.31. SaveJP2Image()

- ***Signature***

Method	public int **SaveJP2Image**(String filename, byte[] image, int width, int height, int pitch, double resX, double resY, int fQuality) throws IBScanException

- ***Description***

  Save JPEG-2000 image to specific file path.

- ***Parameter***

Parameter	Description
filename	File path to save jpeg-2000 image
image	Original image data

width	Width of original image
height	Height of original image
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

1.1.5.32. SaveBitmapImage ()

- ***Signature***

Method	public int **SaveBitmapImage**(String filename, byte[] image, int width, int height, int pitch, double resX, double resY) throws IBScanException

- ***Description***

  Save Bitmap image to specific file path.

- ***Parameter***

Parameter	Description
filename	File path to save Bitmap image
image	Original image data
width	Width of original image
height	Height of original image
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)

1.1.5.33. getEnhancedImageReserved()

- ***Signature***

Method	public void **getEnhancedImageReserved**(String reservedKey, IBScanDevice.ImageData image) throws IBScanException

- ***Description***

  Generate enhanced image from preview, returning extended information.

- ***Parameter***

Parameter	Description
reservedKey	The reserved key to set for the property, as a string
imageData	Input image data which is returned from preview callback

- ***Return***

an array containing information about enhanced image. The contents of the returned array, in order, are

- IBScanDevice.ImageData enhancedImage – enhanced image data
- Integer detectedFingerCount – detected finger count
- IBScanDevice.ImageData[] segmentImageArray – finger array split from the enhanced image data
- IBScanDevice.SegmentPosition[] segmentPositionArray – position data for individual fingers split from enhanced image data

1.1.5.34. Constant field values

Field	Description
MAX_CONTRAST_VALUE	Maximum contrast value. See getContrast(), setContrast().
MIN_CONTRAST_VALUE	Minimum contrast value. See getContrast(), setContrast().
OPTION_AUTO_CAPTURE	Capture option constant for auto capture. See beginCaptureImage().
OPTION_AUTO_CONTRAST	Capture option constant for auto contrast. See beginCaptureImage().
OPTION_IGNORE_FINGER_COUNT	Capture option constant for ignore finger count. See beginCaptureImage().

1.1.5.35. getCombineImage ()

- ***Signature***

Method	public void **getCombineImage**(IBScanDevice.ImageData inImage1, IBScanDevice.ImageData inImage2,int whichHand) throws IBScanException

- ***Description***

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

- ***Parameter***

Parameter	Description
inImage1	Pointer to IBScanDevice.ImageData (index and middle finger)
inImage2	Pointer to IBScanDevice.ImageData (ring and little finger)
whichHand	Information of left or right hand

- ***Return***

  Pointer to IBScanDevice.ImageData (1600 x 1500 fixed size image)

1.1.5.36. getOperableBeeper()

- ***Signature***

Method	public IBScanDevice. BeeperType **getOperableBeeper**() throws IBScanException

- ***Description***

  Get characteristics of operable Beeper on a device.

- ***Return***

  information about Beeper type

1.1.5.37. setBeeper()

- ***Signature***

Method	public void **setBeeper**(BeepPattern bPattern, final long soundTone, final long duration, final long reserved_1, final long reserved_2)

- ***Description***

  Set the value of Beeper on a device.

- ***Parameter***

Parameter	Description
bPattern	Input beep data which is returned from keybuttons callback
soundTone	The frequency of the sound, in specific value. The parameter must be in the range 0 through 2
duration	The duration of the sound, in 25 miliseconds. The parameter must be in the range 1 through 200 at

	ENUM_IBSU_BEEP_PATTERN_GENERIC, in the range 1 through 7 at ENUM_IBSU_BEEP_PATTERN_REPEAT.
reserved_1	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 1 through 8
reserved_2	Reserved, If you set beepPattern to ENUM_IBSU_BEEP_PATTERN_REPEAT reserved_1 can use the operation(start/stop of pattern repeat), 1 to start; 0 to stop

1.1.5.38. getCombineImageEx()

- ***Signature***

Method	public void **getCombineImageEx**(IBScanDevice.ImageData inImage1, IBScanDevice.ImageData inImage2,int whichHand) throws IBScanException

- ***Description***

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

- ***Parameter***

Parameter	Description
inImage1	Pointer to IBScanDevice.ImageData (index and middle finger)
inImage2	Pointer to IBScanDevice.ImageData (ring and little finger)
whichHand	Information of left or right hand

- ***Return***

  an array containing information about result image. The contents of the returned array, in order, are

  - IBScanDevice.ImageData resultImage – result image data
  - IBScanDevice.ImageData[] segmentImageArray – finger array split from the result image data
  - IBScanDevice.SegmentPosition[] segmentPositionArray – position data for individual fingers split from result image data
  - Integer SegmentImageArrayCount – detected segment count

1.1.5.39. generateDisplayImage()

- ***Signature***

Method	public int **generateDisplayImage**(byte[] image, int inWidth, int inHeight, byte[]outImage, int outWidth, int outHeight, byte bkColor, int outFormat, int outQualityLevel, Boolean outVerticalFlip) throws IBScanException

- ***Description***

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

- ***Parameter***

Parameter	Description
image	Original grayscale image data
inWidth	Width of input image
inHeight	Height of input image
outImage	Pointer to buffer that will receive output image. This buffer must hold at least 'outWidth' x 'outHeight' bytes
outWidth	Width of output image
outHeight	Height of output image
bkColor	Background color of output image
outFormat	Image format of output image
outQualityLevel	Image quality of output image, the parameter must be in the range 0 through 2
outVerticalFlip	Enable/disable vertical flip of output image

1.1.5.40. removeFingerImage()

- ***Signature***

Method	public int **removeFingerImage**(long fIndex) throws IBScanException

- ***Description***

  Remove a finger image

- ***Parameter***

Parameter	Description
fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING

1.1.5.41. addFingerImage()

- ***Signature***

Method	public int **addFingerImage**(IBScanDevice.ImageData image, long fIndex, IBScanDevice.ImageType imageType, boolean flagForce) throws IBScanException

- ***Description***

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

- ***Parameter***

Parameter	Description
image	Input image data
fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING
imageType	Image type of input image
flagForce	Indicates whether input image should be saved even if another image is already stored or not. TRUE to be stored force; FALSE to be not stored force

1.1.5.42. isFingerDuplicated()

- ***Signature***

Method	public long **isFingerDuplicated**(IBScanDevice.ImageData image, long fIndex, IBScanDevice.ImageType imageType, int securityLevel) throws IBScanException

- ***Description***

  Checks for the fingerprint duplicate from the stored prints by addFingerImage()

- ***Parameter***

Parameter	Description
image	Input image data
fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING
imageType	Image type of input image

securityLevel	security level for the duplicate checks

- ***Return***

  Bit-pattern matched index variable that will receive result of duplicate

1.1.5.43. isValidFingerGeometry()

- ***Signature***

Method	public long **isValidFingerGeometry**(IBScanDevice.ImageData image, long fIndex, IBScanDevice.ImageType imageType) throws IBScanException

- ***Description***

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

- ***Parameter***

Parameter	Description
image	Input image data
fIndex	Bit-pattern of finger index of input image. ex) IBSU_FINGER_LEFT_LITTLE | IBSU_FINGER_LEFT_RING
imageType	Image type of input image

- ***Return***

  Variable that will receive whether it is valid or not. True to valid; false to invalid.

1.1.5.44. SetEncryptionKey()

- ***Signature***

Method	public int **SetEncryptionKey**(byte[] encyrptionKey, EncyptionMode encMode) throws IBScanException

- ***Description***

  Set encryption key and mode

- ***Parameter***

Parameter	Description
encyrptionKey	input data for encryption key (should be 32 bytes)
encMode	input data for encryption mode. (random, custom)

1.1.6. Interface IBScanListener

1.1.6.1. Summary

Listener for device management events on an IBScan. This listener should be registered by an application with the setScanListener() method.

Special functions are provided for device attachment and detachment and permission status on Android, where the app may be explicitly responsible for soliciting permission.

1.1.6.2. scanDeviceAttached() (Android only)

- ***Signature***

Method	void **scanDeviceAttached**(int deviceId)

- ***Description***

Device attached notification.

- ***Parameter***

Parameter	Description
deviceId	ID of the device. This is the ID that Android assigns to the device, obtained through the UsbDevice getDeviceId() method

1.1.6.3. scanDeviceDetached() (Android only)

- ***Signature***

Method	void **scanDeviceDetached**(int deviceId)

- ***Description***

Device detached notification.

- ***Parameter***

Parameter	Description
deviceId	ID of the device. This is the ID that Android assigns to the device, obtained through the UsbDevice getDeviceId() method

1.1.6.4. scanDevicePermissionGranted() (Android)

- ***Signature***

Method	void **scanDevicePermissionGranted**(int deviceId, boolean granted)

- ***Description***

Device access granted or denied notification. This notification occurs after

requestPermission() has been called. Only scan devices for which permission has been granted can be opened or be described with getDeviceDescription().

- ***Parameter***

Parameter	Description
deviceId	ID of the device. This is the ID that Android assigns to the device, obtained through the UsbDevice getDeviceId() method
granted	true if permission was granted; false if permission was denied

1.1.6.5. scanDeviceCountChanged ()

- ***Signature***

Method	void **scanDeviceCountChanged**(int deviceCount)

- ***Description***

Device count change notification.

- ***Parameter***

Parameter	Description
deviceCount	new count of devices

1.1.6.6. scanDeviceInitProgress ()

- ***Signature***

Method	void **scanDeviceInitProgress**(int deviceIndex, int progressValue)

- ***Description***

Device initialization progress notification. This notification occurs while the openDevice() is executing; or after openDeviceAsync() has been called before initialization completes or an error occurs.

- ***Parameter***

Parameter	Description
deviceIndex	zero-based index of device
progressValue	initialization progress between 0 and 100. A value of 100 indicates that that initialization is complete

1.1.6.7. scanDeviceOpenComplete ()

- ***Signature***

Method	void **scanDeviceOpenComplete**(int deviceIndex, IBScanDevice device, IBScanException exception)

- ***Description***

  Device open complete notification. This notification occurs after openDeviceAsync() has been called when initialization completes or an error occurs.

- ***Parameter***

Parameter	Description
deviceIndex	zero-based index of device
device	opened device, if successful; otherwise, null
exception	exception, if any, encountered while opening device; otherwise, null

1.1.7. Interface IBScanDeviceListener

1.1.7.1. Summary

Listener for scan events on a IBScanDevice. This listener should be registered by an application using the setScanDeviceListener(IBScanDeviceListener) method.

Most of these events occur after beginCaptureImage() has been called. If fingers are touching the platen when the capture is begun, deviceImagePreviewAvailable() will be called immediately and again once no fingers are touching. Periodically, until a final image is achieved, deviceImagePreviewAvailable() will return the current scanner image. Changes in the quantity and quality of finger presses will result in deviceFingerCountChanged() or deviceFingerQualityChanged() calls. If the selected scan type is a rolled finger scan, then deviceAcquisitionBegun() will be called when a flat finger scan has been acquired and the user should begin rolling his or her finger to the left; when the left-roll is complete, deviceAcquisitionCompleted() will be called, and the user should begin rolling back toward the right. When a quality scan with the correct number of fingers (and a full finger roller, if applicable) is available or captureImageManually() is called, deviceImageResultAvailable() or deviceImageResultExtendedAvailable() will supply a final scan image to the application.

1.1.7.2. deviceCommunicationBroken()

- ***Signature***

Method	void **deviceCommunicationBroken**(IBScanDevice device)

- ***Description***

  Communication break notification. This method is called when communication with the device is broken while a capture is in progress.

- ***Parameter***

Parameter	Description

device	device with which communication has been broken

1.1.7.3. deviceImagePreviewAvailable()

- ***Signature***

Method	void **deviceImagePreviewAvailable**(IBScanDevice device, IBScanDevice.ImageData image)

- ***Description***

  Image preview available notification.

- ***Parameter***

Parameter	Description
device	device for which preview image is available
image	preview image

1.1.7.4. deviceFingerCountChanged()

- ***Signature***

Method	void **deviceFingerCountChanged**(IBScanDevice device, IBScanDevice.FingerCountState fingerState)

- ***Description***

  Finger count change notification.

- ***Parameter***

Parameter	Description
device	device for which finger count has changed
fingerState	state of finger count

1.1.7.5. deviceFingerQualityChanged()

- ***Signature***

Method	void **deviceFingerQualityChanged**(IBScanDevice device, IBScanDevice.FingerQualityState[] fingerQualities)

- ***Description***

  Finger quality change notification.

- ***Parameter***

Parameter	Description

device	device for which finger quality has changed
fingerQualities	array of qualities for fingers

1.1.7.6. deviceAcquisitionBegun()

- ***Signature***

Method	void **deviceAcquisitionBegun**(IBScanDevice device, IBScanDevice.ImageType imageType)

- ***Description***

  Device roll acquisition begun notification. If an image type of ROLL_SINGLE_FINGER is being captured, this method will be called when a flat-finger scan has been acquired and the user should begin rolling his or her finger to the left.

- ***Parameter***

Parameter	Description
device	device for which acquisition has begun
imageType	type of image

1.1.7.7. deviceAcquisitionCompleted()

- ***Signature***

Method	void **deviceAcquisitionCompleted**(IBScanDevice device, IBScanDevice.ImageType imageType)

- ***Description***

  Device roll acquisition complete notification. If an image type of ROLL_SINGLE_FINGER is being captured, this method will be called when the left-roll has been completed and the user should begin roller his or her finger to the left to capture the right side of the finger.

- ***Parameter***

Parameter	Description
device	device for which acquisition has completed
imageType	type of image

1.1.7.8. deviceImageResultAvailable()

- ***Signature***

Method	void **deviceImageResultAvailable**(IBScanDevice device,

	IBScanDevice.ImageData image, IBScanDevice.ImageType imageType, IBScanDevice.ImageData[] splitImageArray)

- ***Description***

  Result image available notification.

- ***Parameter***

Parameter	Description
device	device for which result image is available
image	result image data
imageType	type of image
splitImageArray	array of split result image data

1.1.7.9. deviceImageResultExtendedAvailable()

- ***Signature***

Method	void **deviceImageResultExtendedAvailable**(IBScanDevice device, IBScanException imageStatus, IBScanDevice.ImageData image, IBScanDevice.ImageType imageType, int detectedFingerCount, IBScanDevice.ImageData[] segmentImageArray, SegmentPosition[] segmentPositionArray)

- ***Description***

  Result extended image available notification.

- ***Parameter***

Parameter	Description
device	device for which result image is available
imageStatus	status from result image acquisition
image	result image data
imageType	type of image
detectedFingerCount	Detected finger count
segmentImageArray	array of segment result image data
segmentPositionArray	array of segment position data

1.1.7.10. devicePlatenStateChanged()

- ***Signature***

Method	void devicePlatenStateChanged(IBScanDevice device,IBScanDevice.PlatenState platenState)

- ***Description***

  Platen state changed notification. If fingers are touching the platen when a capture is begun, this method will be called immediately and once again when no fingers are touching. Subsequent state changes from touches will not be notified.

- ***Parameter***

Parameter	Description
device	device for which platen state has changed
platenState	new platen state

1.1.7.11. deviceWarningReceived()

- ***Signature***

Method	void devicePlatenStateChanged(IBScanDevice device,IBScanDevice.PlatenState platenState)

- ***Description***

  Warning message notification.

- ***Parameter***

Parameter	Description
device	device for which warning was received
warning	warning received from device

1.1.7.12. devicePressedKeyButtons()

- ***Signature***

Method	void devicePressedKeyButtons (IBScanDevice device,int pressedKeyButtons)

- ***Description***

  Key button notification.

- ***Parameter***

Parameter	Description
device	device for which key button was pressed
pressedKeyButtons	The key button index which is pressed

1.1.8. Nested Classes

1.1.8.1. Class IBScan.DeviceDesc

- ***Description***

  Basic device description structure.

- ***Uses***

  IBScan method getDeviceDesciption()

- ***Fields***

Field	Description
int **deviceId**	ID of the device. This is the ID that Android assigns to the device, obtained through the UsbDevice getDeviceId() method.
String **devRevision**	Device revision
String **fwVersion**	Device firmware version
String **interfaceType**	Device interface type (USB, Firewire)
boolean **isOpened**	Indicates whether device is opened
String **productName**	Device product name
String **serialNumber**	Device serial number
int **spoof**	Check if device supports spoofing

1.1.8.2. Class IBScan.SdkVersion

- ***Description***

  Container to hold version information.

- ***Uses***

  IBScan method getSdkVersion()

- ***Fields***

Field	Description
String **file**	File version string
String **product**	Product version string

1.1.8.3. Enumeration IBScanDevice.EventType

- ***Description***

  Event types.

- ***Uses***

  IBScanDevice method enableEvent()

- ***Values***

Value	Description
COMMUNICATION_BROKEN	Communication with a device is interrupted.
PREVIEW_IMAGE_AVAILABLE	A new preview image is available from a device.
ACQUISITION_BEGUN	Rolled print acquisition when rolling should begin.
ACQUISITION_COMPLETED	Rolled print acquisition when rolling completes.
RESULT_IMAGE_AVAILABLE	Result image is available for a capture.
FINGER_QUALITY_CHANGED	A finger quality changes.
FINGER_COUNT_CHANGED	The finger count changes.
PLATEN_STATE_CHANGED	The platen was not clear when capture started or has since become clear.
WARNING_RECEIVED	A warning message is generated.
RESULT_IMAGE_EXTENDED _AVAILABLE	Result image is available for a capture (with extended information).

1.1.8.4. Enumeration IBScanDevice.FingerCountState

- ***Description***

  Finger count state definitions.

- ***Uses***

  IBScanDeviceListener method deviceFingerCountChanged()

- ***Values***

Value	Description
FINGER_COUNT_OK	Expected number of fingers on platen
TOO_MANY_FINGERS	Too many fingers on platen
TOO_FEW_FINGERS	Too few fingers on platen
NON_FINGER	No fingers on platen

1.1.8.5. Enumeration IBScanDevice.FingerQualityState

- ***Description***

  Finger quality state definitions.

- ***Uses***

IBScanDeviceListener method deviceFingerQualityChanged()

- ***Values***

Value	Description
FAIR	Fair quality
FINGER_NOT_PRESENT	No finger on platen
GOOD	Good quality
POOR	Poor quality
INVALID_AREA_TOP	Finger position is not valid on top side.
INVALID_AREA_LEFT	Finger position is not valid on left side.
INVALID_AREA_RIGHT	Finger position is not valid on right side.

1.1.8.6. Class IBScanDevice.ImageData

- ***Description***

  Container to hold image data together with meta information.

- ***Uses***

  IBScanDeviceListener methods deviceImagePreviewAvailable() and deviceImageResultAvailable(); IBScanDevice method captureImage()

- ***Fields***

Field	Description
short **bitsPerPixel**	Number of bits per pixel
byte[] **buffer**	Byte-array holding image data
IBScanDevice.ImageFormat **format**	Image color format
double **frameTime**	Image acquisition time (in seconds). The value contains the time taken for acquisition from device (excluding processing time).
int **height**	Image vertical size
boolean **isFinal**	Marks image as finally processed. A value of false disqualifies image from further processing, e.g., interim or pre-processed result images
int **pitch**	Image line pitch (in bytes). Positive values indicate top-down line order, negative values indicate bottom-up line order
double **resolutionX**	Horizontal image resolution (in pixels per inch)

double **resolutionY**	Vertical image resolution (in pixels per inch)
int **width**	Image horizontal size)
Int **processThres**	Threshold of image processing

- ***Methods***

Description
public Bitmap **toBitmap**() Create image from image data.
public Bitmap **toBitmapScaled**(int dstWidth, int dstHeight) Create scaled image from the image data.
public Bitmap **toBitmapScaled**(int dstWidth, int dstHeight, IBScanDevice.RollingState rollingState, int rollingLineX) Create scaled image from the image data with rolling line.
public boolean **saveToFile**(java.io.File output, java.lang.String fileFormat) throws java.io.IOException Save image to file.

1.1.8.7. Enumeration IBScanDevice.ImageFormat

- ***Description***

  Image format constants.

- ***Uses***

  IBScanDevice.ImageData nested class

- ***Values***

Value	Description
GRAY	Gray scale image
RGB24	24-bit RGB color image
RGB32	True color RGB image
UNKNOWN	Format not known or set

1.1.8.8. Enumeration IBScanDevice.ImageResolution

- ***Description***

  Image resolution types.

- ***Uses***

IBScanDevice methods isCaptureModeAvailable(), beginCaptureImage().

- ***Values***

Value	Description
RESOLUTION_1000	1000 pixels-per-inch
RESOLUTION_500	500 pixels-per-inch

1.1.8.9. Enumeration IBScanDevice.ImageType

- ***Description***

Supported image types. This is an enumeration of the image types supported by the SDK, and particular scanner models may not supported all types.

- ***Uses***

IBScanDevice methods isCaptureModeAvailable(), beginCaptureImage().

- ***Values***

Value	Description
FLAT_FOUR_FINGERS	Four flat fingers
FLAT_SINGLE_FINGER	Flat single finger
FLAT_TWO_FINGERS	Two flat fingers
ROLL_SINGLE_FINGER	Rolled fingerprint image
TYPE_NONE	Not supported yet

1.1.8.10. Class IBScanDevice.LedState

- ***Description***

Container to hold LED information.

- ***Uses***

IBScanDevice method getOperableLEDs().

- ***Fields***

Field	Description
int **ledCount**	Number of LEDs
IBScanDevice.LedType **ledType**	Type of LEDs
long **operableLEDs**	Bit pattern of operable LEDs

1.1.8.11. Enumeration IBScanDevice.LEOperationMode

- ***Description***

Definitions of light emitting film's operation modes.

- ***Uses***

  IBScanDevice methods getLEOperationMode(), setLEOperationMode().

- ***Values***

Value	Description
AUTO	
OFF	
ON	

1.1.8.12. Enumeration IBScanDevice.PlatenState

- ***Description***

  Platen state definitions.

- ***Uses***

  IBScanDevice method getPlatenStateAtCapture(); IBScanDeviceListener method devicePlatenStateChanged().

- ***Values***

Value	Description
CLEARD	
HAS_FINGERS	

1.1.8.13. Enumeration IBScanDevice.PropertyId

- ***Description***

  General property definitions. Property values of an IBScanDevice are set with setProperty() and gotten with getProperty().

- ***Uses***

  IBScanDevice methods getProperty() and setProperty().

- ***Values***

Value	Description
PRODUCT_ID	(get) Product name string (e.g. "Watson")
SERIAL_NUMBER	(get) Serial number string
VENDOR_ID	(get) Device manufacturer identifier
IBIA_VENDOR_ID	(get) IBIA vendor ID
IBIA_VERSION	(get) IBIA version information

IBIA_DEVICE_ID	(get) IBIA device ID.
FIRMWARE	(get) Firmware version string
REVISION	(get) Device revision string
PRODUCTION_DATE	(get) Production date string
PRODUCT_ID	(get) Product name string (e.g., “Watson”).
PRODUCTION_DATE	(get) Production date string
SERVICE_DATE	(get) Last service date string
IMAGE_WIDTH	(get) Image width value
IMAGE_HEIGHT	(get) Image height value
IGNORE_FINGER_TIME	(get/set) The time in milliseconds to acquire the finger print in the auto capture mode regardless of the number of fingers. The capture option OPTION_AUT_CAPTURE must be given when capture is begun (with beginCaptureImage()). The default value is 4000-ms and the value may range between 2000- and 10000-ms.
RECOMMENDED_LEVEL	(get/set) Auto contrast level value
POLLINGTIME_TO_BGETIMAGE	(get) Polling time for blocking image capture (with captureImage()).
ENABLE_POWER_SAVE_MODE	(get/set) Power save mode. Specify the value “TRUE” to enable or “FALSE” to disable. By default, power save mode is disabled.
RETRY_WRONG_COMMUNICATION	(get/set) The retry count for communication failures. The default value is 6, and the value may range between 1 and 120.
CAPTURE_TIMEOUT	(get/set) The maximum wait time for image capture, in seconds. If -1, the timeout is infinite. Otherwise, the valid range is between 10- and 3600-seconds, inclusive. The default is -1.
ROLL_MIN_WIDTH	(get/set) Minimum distance of rolled fingerprint, in millimeters. The valid range is between 10- and 30-mm, inclusive. The default is 15-mm.
ROLL_MODE	(get/set) roll mode. The valid range is between 0 ~ 1. The default is 1. 0 : no use smear 1 : use notice
ROLL_LEVEL	(get/set) roll level. The valid range is between 0 ~ 2. The default is 1. 0 : low level

	1 : medium level 2 : high level
CAPTURE_AREA_THRESHOLD	(get/set) The area threshold for image capture for flat fingers. The area threshold for beginning rolled finger. The valid range is between 0 and 12, inclusive, with the default of 6.
ENABLE_DECIMATION	(get/set) Enable decimation mode (TRUE to enable or FALSE to disable). Some of devices (or firmware version) does not support this feature.
ENABLE_CAPTURE_ON_RELEASE	(get/set) Enable capture on release (TRUE to enable or FALSE to disable). The default is FALSE. TRUE : the result callback will be called when user release the finger from the sensor. FALSE : the result callback will be called when the quality of finger become good
DEVICE_INDEX	(get) The device index
DEVICE_ID	(get) The device ID which has same information with UsbDevice class of Android
SUPER_DRY_MODE	(get/set) Some of devices (or firmware version) does not support this feature. The default is FALSE. TRUE : Enable dry mode. FALSE : Disable dry mode
MIN_CAPTURE_TIME_IN_SUPER_DRY_MODE	(get/set) It is a minimum capture time when the dry mode is enabled with the property ENUM_IBSU_PROPERTY_SUPER_DRY_MODE. Some of devices (or firmware version) does not support this feature. The valid range is between 600- and 3000-ms, inclusive, with the default of 2000-ms.
ROLLED_IMAGE_WIDTH	(get) Rolled image width value
ROLLED_IMAGE_HEIGHT	(get) Rolled image height value
NO_PREVIEW_IMAGE	(get/set) Enable the drawing for preview image
ROLL_IMAGE_OVERRIDE	(get/set) Enable to override roll image
WARNING_MESSAGE_INVALID_AREA	(get/set) Enable the warning message for invalid area for result image
ENABLE_WET_FINGER_DETECT	(get/set) Enable wet detect function
WET_FINGER_DETECT_LEV	(get/set) Change wet detect level. The valid range is between 1 and 5. The default is 3.

EL	1 : Lowest level for detect wet finger : less sensitive 5 : Highest level for detect wet detect : more sensitive
WET_FINGER_DETECT_LEVEL_THRESHOLD	(get/set) Change threshold for each wet detect level. The valid range is between 10 and 1000. The default is "50 100 150 200 250" 50 : Threshold of lowest level for detect wet finger 250 : Threshold of highest level for detect wet finger
START_POSITION_OF_ROLLING_AREA	(get/set) Control rolling area vertically. The valid range is between 0 and 9. The default is 0. 0 : minimum position 9 : maximum position
START_ROLL_WITHOUT_LOCK	(get/set) Enable rolling without lock. The default is FALSE.
ENABLE_TOF	(get/set) Enable TOF function. The default is set depending on the devices.
ENABLE_ENCRYPTION	(get/set) Enable Encryption for capture images The default is FALSE
IS_SPOOF_SUPPORTED	(get) Check if the device support spoof function or not
ENABLE_SPOOF	(get/set) Enable spoof function The default is FALSE.
SPOOF_LEVEL	(get/set) Change spoof level. The valid range is between 0 and 10. The default is 5. [Get and set.] 0 : Lowest level for spoof finger : less sensitive 10 : Highest level for spoof finger : more sensitive
VIEW_ENCRYPTION_IMAGE_MODE	(get/set) View encrypt Image The default is FALSE.
FINGERPRINT_SEGMENTATION_MODE	(get/set) Select fingerprint segmentation mode The default is 0.
RESERVED_1 (200) RESERVED_2 (201) RESERVED_100 (202)	Reserved for manufacturer strings. [Need a reserve code]
RESERVED_IMAGE_PROCESS_THRESHOLD	(get/set) The previmage processing threshold. [Need a partner or reserve code] The valid range is between 0 and 2, inclusive, with

	the default of 0 on embedded processor (ARM, Android and Windows Mobile), and with the default of 2 on PC. 0 : IMAGE_PROCESS_LOW 1 : IMAGE_PROCESS_MEDIUM 2 : IMAGE_PROCESS_HIGH
RESERVED_ENABLE_TOF_FOR_ROLL	(get/set) Enable TOF for roll capture The default is FALSE
RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_FLAT	(get/set) Change brightness threshold for flat capture. The default values are depending on the scanner.
RESERVED_CAPTURE_BRIGHTNESS_THRESHOLD_FOR_ROLL	(get/set) Change brightness threshold for roll capture. The default values are depending on the scanner.
RESERVED_ENHANCED_RESULT_IMAGE	(get/set) Change result image to be enhanced The default values are FALSE.

1.1.8.14. Class IBScanDevice.RollingData

- ***Description***

  Rolling state data.

- ***Uses***

  IBScanDevice method getRollingState().

- ***Fields***

Field	Description
int **rollingLineX**	Horizontal position of the vertical rolling line.
IBScanDevice.RollingState **rollingState**	The rolling state

1.1.8.15. Enumeration IBScanDevice.RollingState

- ***Description***

  Rolling state definitions.

- ***Uses***

  IBScanDevice.RollingData nested class

- ***Values***

Value	Description
COMPLETE_ACQUISITION	Acquisition of scan for roll has completed

NOT_PRESENT	Acquisition has not begun
RESULT_IMAGE	A result image is already available
TAKE_ACQUISITION	Acquisition of scan for roll is occurring

1.1.8.16. Class IBScanDevice.SegmentPosition

- ***Description***

  Segment position.

- ***Uses***

  IBScanDevice method captureImageExtended() and IBScanDeviceListener method deviceImageResultExtendedAvailable().

- ***Fields***

Field	Description
int **x1**	x-coordinate of the first vertex
int **y1**	y-coordinate of the first vertex
int **x2**	x-coordinate of the second vertex
int **y2**	y-coordinate of the second vertex
int **x3**	x-coordinate of the third vertex
int **y3**	y-coordinate of the third vertex
int **x4**	x-coordinate of the fourth vertex
int **y4**	y-coordinate of the fourth vertex

1.1.8.17. Enumeration IBScanDevice.CombineImageWhichHand

- ***Description***

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

- ***Uses***

  IBScanDevice method getCombineImage()

- ***Fields***

Field	Description
COMBINE_IMAGE_LEFT_HAND	Left Hand Image.
COMBINE_IMAGE_RIGHT_HAND	Right Hand Image.

1.1.8.18. Enumeration IBScanDevice.EncyptionMode

- ***Description***

Enumeration of Encryption mode.

- ***Uses***

  IBScanDevice method SetEncryptionKey()

- ***Fields***

Field	Description
ENCRYPTION_KEY_RANDOM	Random Key generated by own library.
ENCRYPTION_KEY_CUSTOM	Custom Key provided by user.

1.1.9. Common Nested Classes

1.1.9.1. Class IBCommon.ImageDataExt

- ***Description***

  Container to hold image data together with extended meta data.

- ***Uses***

  IBScanDevice method getResultImageExt()

- ***Fields***

Field	Description
ImageFormat **imageFormat**	
ImpressionType **impressionType**	
FingerPosition **fingerPosition**	
CaptureDeviceTechId **captureDeviceTechId**	
short **captureDeviceVendorId**	
short **captureDeviceTypeId**	
short **scanSamplingX**	
short **scanSamplingY**	
short **ImageSamplingX**	
short **imageSamplingY**	
short **imageSizeX**	
short **imageSizeY**	
byte **scaleUnit**	
byte **bitDepth**	
byte[] **imageData**	

1.1.9.2. Enumeration IBCommon.ImageFormat

- ***Description***

  Image formats.

- ***Uses***

  IBCommon.ImageDataExt member imageFormat

- ***Values***

Value	Description
NO_BIT_PACKING	
BIT_PACKED	
WSQ	
JPEG_LOSSY	
JPEG2000_LOSSY	
JPEG2000_LOSSLESS	
PNG	

1.1.9.3. Enumeration IBCommon.ImpressionType

- ***Description***

  Impression type.

- ***Uses***

  IBCommon.ImageDataExt member impressionType

- ***Values***

Value	Description
LIVE_SCAN_PLAIN	
LIVE_SCAN_ROLLED	
NONLIVE_SCAN_PLAIN	
NONLIVE_SCAN_ROLLED	
LATENT_IMPRESSION	
LATENT_TRACING	
LATENT_PHOTO	
LATENT_LIFT	
LIVE_SCAN_SWIPE	

LIVE_SCAN_VERTICAL_ROLL	
LIVE_SCAN_PALM	
NONLIVE_SCAN_PALM	
LATENT_PALM_IMPRESSION	
LATENT_PALM_TRACING	
LATENT_PALM_PHOTO	
LATENT_PALM_LIFT	
LIVE_SCAN_OPTICAL_CONTRCTLESS_ PLAIN	
OTHER	
UNKNOWN	

1.1.9.4. Enumeration IBCommon.FingerPosition

- ***Description***

  Finger positions.

- ***Uses***

  IBCommon.ImageDataExt member fingerPosition; IBScanDevice function getResultImageExt()

- ***Values***

Value	Description
UNKNOWN	
RIGHT_THUMB	
RIGHT_INDEX_FINGER	
RIGHT_MIDDLE_FINGER	
RIGHT_RING_FINGER	
RIGHT_LITTLE_FINGER	
LEFT_THUMB	
LEFT_INDEX_FINGER	
LEFT_MIDDLE_FINGER	
LEFT_RING_FINGER	

LEFT_LITTLE_FINGER	
PLAIN_RIGHT_FOUR_FINGERS	
PLAIN_LEFT_FOUR_FINGERS	
PLAIN_THUMBS	
UNKNOWN_PALM	
RIGHT_FULL_PALM	
RIGHT_WRITERS_PALM	
LEFT_FULL_PALM	
LEFT_WRITERS_PALM	
RIGHT_LOWER_PALM	
RIGHT_UPPER_PALM	
RIGHT_OTHER	
LEFT_OTHER	
RIGHT_INTERDIGITAL	
RIGHT_THENAR	
RIGHT_HYPOTHENAR	
LEFT_INTERDIGITAL	
LEFT_THENAR	
LEFT_HYPOTHENAR	
RIGHT_INDEX_AND_MIDDLE	
RIGHT_MIDDLE_AND_RING	
RIGHT_RING_AND_LITTLE	
LEFT_INDEX_AND_MIDDLE	
LEFT_MIDDLE_AND_RING	
LEFT_RING_AND_LITTLE	
RIGHT_INDEX_AND_LEFT_INDEX	
RIGHT_INDEX_AND_MIDDLE_AND_RING	
RIGHT_MIDDLE_AND_RING_AND_LITTLE	
LEFT_INDEX_AND_MIDDLE_AND_RING	
LEFT_MIDDLE_AND_RING_AND_LITTLE	

1.1.9.5. Enumeration IBCommon.CaptureDeviceTechId

- ***Description***

Capture device technology ID.

- ***Uses***

  IBCommon.ImageDataExt member captureDeviceTechId

- ***Values***

Value	Description
UNKNOWN_OR_UNSPECIFIED	
WHITE_LIGHT_OPTICAL_TIR	
WHITE_LIGHT_OPTICAL_DIRECT_VIEW_ ON_PLATEN	
WHITE_LIGHT_OPTICAL_TOUCHLESS	
MONOCHROMATIC_VISIBLE_OPTICAL_TIR	
MONOCHROMATIC_VISIBLE_OPTICAL_ DIRECT_VEIW_ON_PLATEN	
MONOCHROMATIC_VISIBLE_OPTICAL_ TOUCHLESS	
MULTISPECTRAL_OPTICAL_TIR	
MULTISPECTRAL_OPTICAL_DIRECT_VIEW_ ON_PLATEN	
MULTISPECTRAL_OPTICAL_TOUCHLESS	
ELECTRO_LUMINESCENT	
SEMICONDUCTOR_CAPACITIVE	
SEMICONDUCTOR_RF	
SEMICONDUCTOR_THEMAL	
PRESSURE_SENSITIVE	
ULTRASOUND	
MECHANICAL	
GLASS_FIBER	

Support Contact Information:

www.integratedbiometrics.com

Integrated Biometrics, LLC

North American Office

Physical Address for Package Delivery
121 Broadcast Drive
Spartanburg SC 29303

For Mailings & Correspondence
PO Box 170938
Spartanburg, SC 29301

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

Sales & Pricing Inquiries
sales@integratedbiometrics.com

Terms & Conditions of a Sale

Terms & Conditions for Supplier Purchases

Sales Administration
marci.bowers@integratedbiometrics.com

Technical Support
technical@integratedbiometrics.com

South Korean Office

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

Phone
+82-31-777-2207

Sales Administration
everun@ibkr.co.kr