

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

SMUFS-Android Application Program Interface (API)

Version 0.3

SMUFS Biometric Solutions Ltd.

www.smufsbio.com

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

Revision History

| Date | Version | Description | Author |
|----------|---------|---|--------------------|
| 28/08/14 | 0.1 | Initial draft | rishi@smufsbio.com |
| 09/09/14 | 0.2 | Adding Auto connect back feature, related properties and few new methods like controlling sensor power. | rishi@smufsbio.com |
| 19/02/15 | 0.3 | Added new functions and updated device properties with battery details | rishi@smufsbio.com |

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 5 |
| 1.1 | Specification Objectives | 5 |
| 1.2 | Intended Audiences..... | 5 |
| 1.3 | Prerequisites..... | 5 |
| 1.4 | Configuring custom Android Studio project with SMUFS SDK | 5 |
| 2 | Public Classes | 7 |
| 2.1 | SmufsDevice | 7 |
| 2.2 | FingerImage | 7 |
| 2.3 | DeviceProperties | 7 |
| 3 | SmufsDevice | 8 |
| 3.1 | initialize | 9 |
| 3.2 | getConnectionError | 10 |
| 3.3 | getFingerScan | 11 |
| 3.4 | cancelAsyncScan | 12 |
| 3.5 | getSDKVersion | 12 |
| 3.6 | getProperties | 12 |
| 3.7 | StartSensorPower | 13 |
| 3.8 | CloseSensorPower | 13 |
| 3.9 | setDeviceIdleTimeout..... | 14 |
| 3.10 | setDeviceConnectedTimeout | 14 |
| 3.11 | setAutoConnectFeature | 15 |
| 3.12 | setDeviceAutoConectCount | 15 |
| 3.13 | setDeviceAutoConnectFrequency | 16 |
| 3.14 | setDeviceAutoShutdown | 16 |
| 3.15 | setDevicePassword..... | 17 |
| 3.16 | disableDevicePassword | 17 |
| 3.17 | sleep | 18 |
| 3.18 | wakeup | 18 |
| 3.19 | disconnect | 18 |
| 4 | FingerImage..... | 20 |
| 4.1 | ImageCaptureStatus property | 20 |
| 4.2 | getBitmapImage | 20 |
| 4.3 | getWSQImage..... | 20 |
| 4.4 | getGrayScaleImageData | 21 |
| 4.5 | getImageHeight | 21 |
| 4.6 | getImageWidth | 21 |
| 4.7 | isValidImage | 22 |
| 5 | DeviceProperties | 23 |
| 5.1 | DeviceIdleTimeOutSec..... | 23 |

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

| | | |
|----------|---------------------------------------|-----------|
| 5.2 | DeviceConnectedTimeOutSec | 23 |
| 5.3 | BatteryCutOff_Percentage | 23 |
| 5.4 | DeviceVersion0 | 23 |
| 5.5 | DeviceVersion1 | 23 |
| 5.6 | ProductID | 23 |
| 5.7 | BootloaderVersion0 | 23 |
| 5.8 | BootloaderVersion1 | 24 |
| 5.9 | ProductSerialNumber | 24 |
| 5.10 | MAC_Address | 24 |
| 5.11 | manufacturerString | 24 |
| 5.12 | usbSerialNumber | 24 |
| 5.13 | deviceReleaseNumber | 24 |
| 5.14 | deviceVID | 24 |
| 5.15 | devicePID | 24 |
| 5.16 | AutoConnectBackFeature | 24 |
| 5.17 | AutoConnectBackFrequency | 24 |
| 5.18 | AutoConnectBackAttemptCount | 25 |
| 5.19 | Current_ConnectBack_MAC_Address | 25 |
| 5.20 | connectbackDeviceType | 25 |
| 5.21 | IsSensorPowered | 25 |
| 5.22 | ConnectionPasswordEnabled | 25 |
| 5.23 | stateOfCharge | 25 |
| 5.24 | Voltage | 25 |
| 5.25 | Current | 25 |
| 5.26 | Temprature | 26 |
| 5.27 | AbsoluteRemainingChargeValue | 26 |
| 5.28 | powerSource | 26 |
| 5.29 | batteryState | 26 |
| 5.30 | IsCharging | 26 |
| 6 | Tutorials | 27 |
| 6.1 | SmufsCaptureSample | 27 |

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

1 Introduction

SMUFS-Android API provides an easy-to-use software library to communicate and handle the SMUFS mobile biometric products with Android-based devices.

1.1 Specification Objectives

The objective of the API is to formally document all application programmer interfaces available in the library as well as to provide samples and explanation for easy integration.

1.2 Intended Audiences

The API has the following intended audiences:

- Architecture Team
- Development Team
- Test Team

1.3 Prerequisites

- Android Studio is required for all future Android apps development.
- You need to download the latest Android Studio, you can download it from:
- <http://developer.android.com/sdk/index.html#top>

1.4 Configuring custom Android Studio project with SMUFS SDK

- In the Module libs folder copy the provided library file ie: btsdk-release.aar. You can copy this file from redistributable folder of SDK package.

- Modify Module build.gradle file as follows:

- Add repositories group:

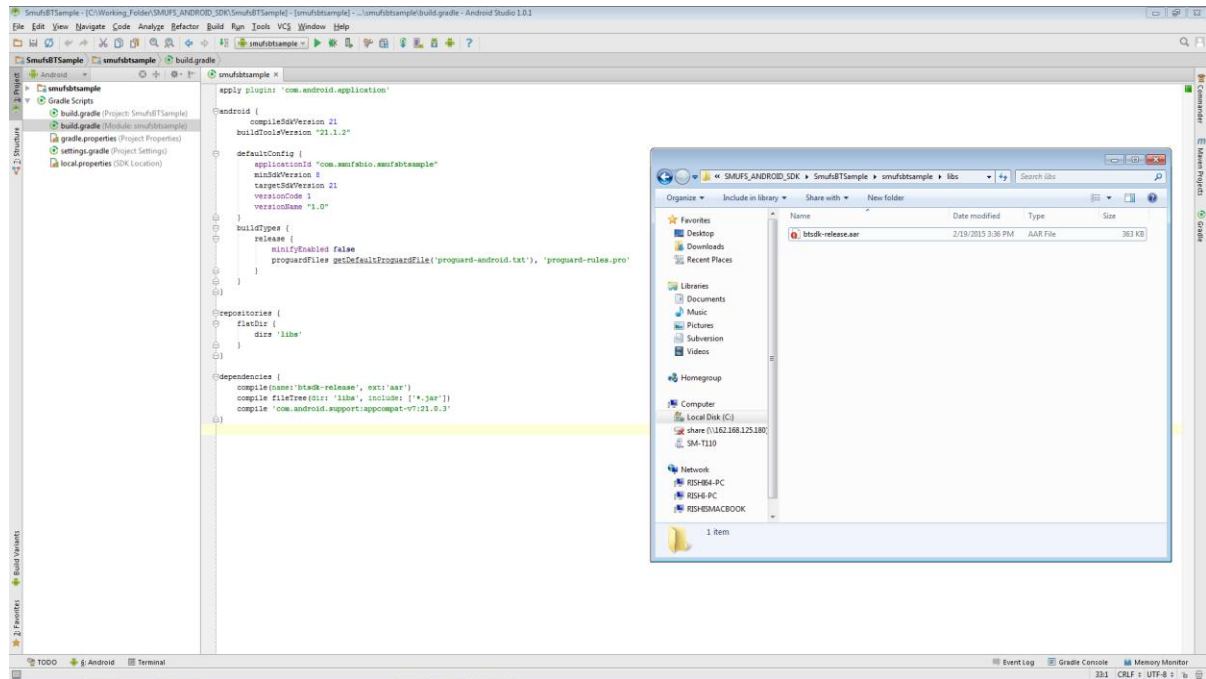
```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

- Then add SDK dependency in dependencies group:

```
dependencies {
    compile(name:'btsdk-release', ext:'aar')
    ....
}
```

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

Please refer to the following picture for configuring custom application Android Studio project:



| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

2 Public Classes

Smuf's Android SDK exposes the following public classes:

2.1 SmufsDevice

It's the main communication class to connect and talk to device. It provides all the required methods to get/set device properties and capture image scan.

2.2 FingerImage

This class holds the captured finger scan status, finger images in all the formats supported by this SDK (currently WSQ and BMP) and its associated properties.

2.3 DeviceProperties

This class holds all the device properties like battery data, timeouts, device access password setting etc

Imports Required to access these classes

```
import com.smufsbio.btsdk.DeviceProperties;
import com.smufsbio.btsdk.FingerImage;
import com.smufsbio.btsdk.SmufsBTDevice;
```

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3 SmufsDevice

This is the main class providing all the communication methods with the Smuf's device. This section defines all the methods and properties provided by this class. Typical usage case will be as follows:

- Initialize the SmufsDevice object by invoking **initialize** method.
- Get all device properties by invoking **getProperties** method.
- Start the sensor power by invoking **StartSensorPower** method.
- Capture the finger scans by invoking **getFingerScan** method.
- Close the sensor power by invoking **CloseSensorPower** method.
- When done with the device close the device connection by **disconnect** method.

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.1 initialize

This method initializes the SDK and tries to connect to the desired Smufs Device. This should be the first method which user application. If it successfully connects to the desired device it returns the SmufsDevice object otherwise it returns null.

There is no way to detect if device is password protected or not before connecting to it. So first you can try with null and if it fails with INVALID_PASSWORD then you can prompt the user to capture the device password.

Typical usage will be:

```
_dev = SmufsBTDevice.initialize(devName,password,deviceDisconnectionHandler);
```

initialize

```
public static synchronized SmufsBTDevice initialize(String deviceName, String devicePassword, Handler deviceDisconnectionStatusCallback)
```

Initializes and connects to SMUFS Bluetooth device

Parameters:

- deviceName – SMUFS Bluetooth device name to connect to
- devicePassword – device password if device is password protected else null
- deviceDisconnectionStatusCallback – Connection status callback handler, where SDK delivers the device connection and disconnection notifications

Returns:

- SmufsDevice object for communicating with the device or null if connection failed

ConnectionState Callback Signature

```
private Handler deviceDisconnectionHandler = new Handler()
{
    @Override public void handleMessage(android.os.Message msg) {
        try {
            //Application handles device disconnection and connect back events
            SmufsBTDevice.DEVICE_CONNECTION_STATUS state;
            state = (SmufsBTDevice.DEVICE_CONNECTION_STATUS) msg.obj;
            //state is DEVICE_CONNECTION_STATUS enumeration
            .... }}
}
```

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.2 getConnectionError

This method returns the details of most recent connection attempt result. You should use this method if initialize call returns null to get the reason of failure.

getConnectionError

```
public static Connection_Errors getConnectionError()
Returns the connection error code.
```

```
public enum Connection_Errors {
    CONNECTION_SUCCESSFULL(0),
    DEVICE_NOT_CONNECTED(1),
    INVALID_FIRMWARE_VERSION(2),
    COMMUNICATION_ERROR(3),
    INVALID_PASSWORD(4),
    SESSION_ALREADY_EXISTS(5),
    UNKNOWN(6);
}
```

Typical Usage

```
_dev = SmufsBTDevice.initialize(devName,password,deviceDisconnectionHandler);

if(_dev != null)
{
    Log.d("CONNCLASS","Failed to connect to " + devName + " with err code :"+
    SmufsBTDevice.getConnectionError().toString());
}
```

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.3 getFingerScan

The method captures the finger scan from the device. Before calling this method sensor needs to be powered, you can power up the sensor by StartSensorPower method. If you will try to capture image without powering up the sensor first it will fail with SENSOR_NO_POWER but then it will start the sensor power. Powering up the sensor takes around 1.6 seconds so second call to getFingerScan will return you the image. Based on the parameters it can capture synchronously or asynchronously. In the case of asynchronous mode user has to provide the callback method of their application where SDK can deliver the capture status messages. Please refer to the sample code in the Android Sample application folder.

In the case of asynchronous scan you need to provide callback of the type Handler and you will get FingerImage object in message.obj and message.arg1 will have the receivedBytes and message.arg2 will have the totalBytes.

Status of the finger scans and finger image can be retrieved from the [Finger Image class](#).

FingerImage Callback Method

```
private Handler AsyncScanCallback = new Handler() {
    @Override public void handleMessage(android.os.Message msg) {
        try {
            FingerImage objFingerImage = (FingerImage) msg.obj;
            int receivedBytes = msg.arg1;
            int totalBytes = msg.arg2;
        }
    }
}
```

getFingerScan

```
public synchronized FingerImage getFingerScan(boolean aSync,int timeoutms,Handler scanProgresscallback)
```

Commands the scanner to capture the finger scan and return to the application synchronously or asynchronously.

Parameters:

AsyncScan – Set to true if you want asynchronous capture and false for synchronous scan.

timeout – Timeout to wait for finger scan in seconds.

scanProgresscallback – Application defined callback method for asynchronous scan updates or null for synchronous call.

Returns:

- FingerImage Object Containing image data, capture status messages and image properties

See Also:

- **FingerImage**

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.4 **cancelAsyncScan**

This method cancels the finger scan process if device is currently capturing. It's a Synchronous method, which blocks the control until it closes the scan process.

cancelAsyncScan

```
public void cancelFingerScan()
```

Cancels the scanning process if currently capturing asynchronously.

3.5 **getSDKVersion**

This method returns the SDK version of the Android Smufs SDK.

Version format is : MSB_B2_B1_LSB : MSB = MAJOR, Byte2 = MINOR and BYTE1 & LSB is 16 bit revision number.

getSDKVersion

```
public int getSDKVersion()
```

Returns : The SDK version number in integer

3.6 **getProperties**

This method fetches all the device properties. Please refer to [DeviceProperties](#) Class for detailed properties.

getProperties

```
public synchronized DeviceProperties getProperties()
```

Fetches the connected SMUFS device properties

Returns:

- DeviceProperties Object containing SMUFS device information, null if there is any error.

See Also:

- [DeviceProperties](#)

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.7 StartSensorPower

This method will switch ON the sensor power if sensor is in deep sleep state. It can help your application to start the device power whenever really needed. It will return true immediately after instructing the device to start sensor power. On the device side it will take around 1.6 seconds to put the sensor in active state and ready to be used. After invoking this method you should make sure whether the sensor is ready or not by retrieving the device properties and checking for [IsSensorPowered](#) before invoking [getFingerScan](#) function. It's better to call [getProperties](#) at least after 1.5+ seconds as device will be busy powering up the sensor. If you will start calling [getProperties](#) earlier then it will delay the powering up process.

Alternatively you can also switch ON the sensor power by calling [getFingerScan](#) method directly, if sensor is in deep sleep and device gets a finger scan request, device will return with SENSOR_NO_POWER and will start the sensor power immediately so that device can be ready for the next finger scan request. Even in this case you should check [IsSensorPowered](#) after around 1.5+ seconds to make sure power is up before calling [getFingerScan](#) again.

StartSensorPower

```
public synchronized boolean StartSensorPower()
Switches ON the sensor power and places it in active state.
Returns:
• true if successful else false
```

Typical usage

```
_dev.StartSensorPower();
sleep(1500);
DeviceProperties dp = _dev.getDeviceProperties();
While(!dp.IsSensorPowered){
    Sleep(200);
    dp = _dev.getDeviceProperties();
}
```

3.8 CloseSensorPower

This method places the sensor in deep sleep state, which can save lot of battery power. In your application you should try to use this method when you are sure your application don't need scans for the next few minutes or so. Powering up sensor back will take around 1.6 seconds so optimize the battery power usage as applicable by the application scenario.

CloseSensorPower

```
public synchronized boolean CloseSensorPower()
Switches OFF the sensor power and places it in deep sleep state.
Returns:
• true if successful else false
```

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.9 **setDeviceIdleTimeout**

This method sets the auto shutdown idle timeout in seconds for the device, when the device is in Bluetooth discoverable mode and it's idle for this given timeout period it will auto shutdown itself if running on battery. Valid range is 180 seconds (3 minutes) to 900 seconds (15 minutes). To disable the auto idle timeout feature you can set it to 0.

setDeviceIdleTimeout

```
public synchronized boolean setDeviceIdleTimeout(int timeOutsecs)
```

Sets the device idle timeout in seconds.

Returns:

- true if successful else false

3.10 **setDeviceConnectedTimeout**

This method sets the auto shutdown connected timeout in seconds for the device, when the device is in Bluetooth connected mode and there is no communication on the connected channel for this given timeout period it will auto shutdown itself only if running on battery. Valid range is 180 seconds (3 minutes) to 900 seconds (15 minutes). To disable the auto connected timeout feature you can set it to 0.

setDeviceConnectedTimeout

```
public synchronized boolean setDeviceConnectedTimeout(int timeOutsecs)
```

Sets the device connected timeout in seconds.

Returns:

- true if successful else false

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.11 **setAutoConnectFeature**

This method switches ON the Auto Connect feature. If SDK loses Bluetooth connection during normal operation it immediately starts trying to connect back to the connected device for DeviceProperty.AutoConnectCount times at DeviceAutoConnectFrequency rate.

setAutoConnectFeature

```
public synchronized boolean setAutoConnectFeature(boolean enabled)
```

Sets the device auto connect feature On or Off.

Parameters:

enabled – Set to true to enable AutoConnectFeature and set to false to disable.

Returns:

- true if successful else false

3.12 **setDeviceAutoConectCount**

This method sets the number of connection attempts while SDK is trying to connect back. Valid Range is 1 to 50. If SDK cannot connect back either when Smufs device is not switched on or is out of range then after these many attempts SDK will stop trying.

setDeviceAutoConectCount

```
public synchronized boolean setDeviceAutoConnectCount(int count)
```

Sets the device auto connect back count

Parameters:

count – integer from 1 to 50.

Returns:

- true if successful else false

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.13 **setDeviceAutoConnectFrequency**

This is the time duration SDK waits while in auto connect mode before attempting the next auto connect. Valid Range is 1 to 5 seconds.

setDeviceAutoConnectFrequency

```
public synchronized boolean setDeviceAutoConnectFrequency(int frequencysecs)
```

Sets the device auto connect frequency in seconds

Parameters:

frequencysecs – integer from 1 to 5.

Returns:

- true if successful else false

3.14 **setDeviceAutoShutdown**

This method will set the battery threshold in percentage at which device should auto shutdown to protect the battery reaching drained state. Valid range is 5% to 20%

setDeviceAutoShutdown

```
public synchronized boolean setDeviceAutoShutdown(int batteryPercent)
```

Sets the device battery percentage threshold to enable auto shutdown.

Parameters:

batteryPercent – integer from 5 to 20.

Returns:

- true if successful else false

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.15 **setDevicePassword**

This method enables the device access password. Once this password is set on the device, every initialize call will require this password or device connection will fail with Connection Error code INVALID_PASSWORD. Valid range is 6 to 16 characters.

setDevicePassword

```
public synchronized boolean setDevicePassword(String devicePassword)
```

Sets the device access password.

Parameters:

devicePassword – String from 6 to 16 characters.

Returns:

- true if successful else false

3.16 **disableDevicePassword**

This method disables the device access password. You have to provide the valid password previously set through setDevicePassword to disable the access password. Valid range is 6 to 16 characters.

disableDevicePassword

```
public synchronized boolean disableDevicePassword(String devicePassword)
```

Sets the device access password.

Parameters:

devicePassword – String from 6 to 16 characters.

Returns:

- true if successful else false

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

3.17 sleep

This method puts the SDK in sleep state. Actually SDK closes its active channel, saves the context and stop its all execution threads. This method is helpful when application has to manage going to background eg: when user presses the Home button, call, SMS interruption. Application should call this method to put SDK to sleep.

sleep

-(BOOL)sleep;

Puts the SDK to sleep state.

Returns:

- true if successful else false

3.18 wakeup

This method wakes up the SDK from sleep state. In this method SDK attempts to connect back to the Smufs Device which was connected at the time of sleep with same parameters (like device password). This method is helpful when application has to manage coming to foreground state. Application should wakeup when any activity is starting.

You can call wake up in synchronous or asynchronous mode. In asynchronous mode SDK will call back the provided handler with Connection Status message

wakeup

public synchronized boolean wakeup(Handler deviceDisconnectionStatusCallback)

Wakes up the SDK from sleep state.

Parameters:

deviceDisconnectionStatusCallback – Null if using synchronous mode else the Handler which will receive wakeup status.

Returns:

- true if successful else false

3.19 disconnect

This method disconnects from the connected Smuf's device and releases all the associated resources. This method should be the last method which user application calls after user application is done with the device. Optionally you can request device to shut down if it's running on battery.

disconnect

public synchronized boolean disconnect(boolean shutdown)

Disconnects from the connected device.

Parameters:

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

shutdownDevice : set to true to request device to shut down if running on battery else false
Returns: true if successful else false

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

4 FingerImage

4.1 ImageCaptureStatus property

ImageCaptureStatus property returns the current status of finger scan.

Values for ImageCaptureStatus can be as follows:

- GRABIMAGE_FAILED
- FINGER_NOT_ON_SENSOR
- SENSOR_NO_POWER
- CAPTURING_ASYNC
- IMAGE_CAPTURED
- ERROR_UNKNOWN
- INVALID_CALLBACK_PROVIDED
- FINGERIMAGE_TIMEDOUT
- ALREADY_CAPTURING
- USER_CANCELED_CAPTURE
- DEVICE_DISCONNECTED
- COMPRESSING_WSQ

4.2 getBitmapImage

This method returns the android.graphics Bitmap class object for the last captured image. It returns the image as white on black.

getBitmapImage

```
public Bitmap getBitmapImage()
```

Gets the RAW bitmap image

Returns:

- Bitmap of the grayscale image

4.3 getWSQImage

This method returns the byte buffer filled with WSQ image data for the last captured image.

getWSQImage

```
public byte[] getWSQImage()
```

Gets the WSQ image data

Returns:

- Byte array of the wsq image data

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

4.4 **getGrayScaleImageData**

This method returns the byte array for the last captured image or “null” if image is not valid or other problem.

getGrayScaleImageData

```
public byte[] getGrayScaleImageData()
Gets the RAW Gray Scale Image
Returns:


- Byte Array of the grayscale image

```

4.5 **getImageHeight**

This method returns the Height for the last captured image in pixels.

getImageHeight

```
public int getImageHeight()
Gets the image height in pixels
Returns:


- The number of pixels of the image height

```

4.6 **getImageWidth**

This method returns the Width for the last captured image in pixels.

getImageWidth

```
public int getImageWidth()
Gets the image width in pixels
Returns:


- The number of pixels of the image width

```

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

4.7 isValidImage

This method returns the validity status of image. Either true if image is valid or false otherwise.

isValidImage

```
public boolean isValidImage()
```

Gets the image validity status

Returns:

- True if the image is valid, otherwise false

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

5 DeviceProperties

This class holds all the Properties of a connected Smuf's device.

5.1 DeviceIdleTimeOutSec

This integer property is auto shutdown idle timeout in seconds for the device, when the device is in Bluetooth discoverable mode and it's idle for this given timeout period then it will auto shutdown itself only if running on battery. 0 means never auto shutdown. Default is 180 seconds (3 minutes).

5.2 DeviceConnectedTimeOutSec

This integer property is auto shutdown connected timeout in seconds for the device, when the device is in Bluetooth connected mode and there is no communication on the connected channel for this given timeout period it will auto shutdown itself only if running on battery. 0 means never auto shutdown. Default is 180 seconds (3 minutes).

5.3 BatteryCutOff_Percentage

This integer property is battery threshold in percentage at which device should auto shutdown to protect the battery reaching drained state. Default is 5%. Device will auto shutdown only if running on Battery.

5.4 DeviceVersion0

This byte property returns the Device major firmware version.

5.5 DeviceVersion1

This byte property returns the Device minor firmware version.

5.6 ProductID

This byte property returns the Product Id of the device. Product IDs for the Smuf's devices are as follows:

- SMUFS BTWSQ, Product ID = 16

5.7 BootloaderVersion0

This byte property returns the Device Boot loader major firmware version.

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

5.8 **BootloaderVersion1**

This byte property returns the Device Boot loader minor firmware version.

5.9 **ProductSerialNumber**

This string property returns the unique serial number of the device.

5.10 **MAC_Address**

This string property returns the device Bluetooth MAC address.

5.11 **manufacturerString**

This string property returns the Manufacturer String.

5.12 **usbSerialNumber**

This string property returns the USB serial number.

5.13 **deviceReleaseNumber**

This integer property returns the device Release Number.

5.14 **deviceVID**

This integer property returns USB device Vendor ID.

5.15 **devicePID**

This integer property returns USB device Product ID.

5.16 **AutoConnectBackFeature**

This Boolean property returns whether Auto connect back feature is switched on or not.

5.17 **AutoConnectBackFrequency**

This integer property has the time duration device waits while in auto connect mode before attempting the next auto connect. This time can be very crucial as Smuf's device will get a chance to accept any new incoming pairing or connection requests from other BT devices. Default is 1.

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

5.18 **AutoConnectBackAttemptCount**

This integer property holds the number of connection attempts that Smuf's device will try to connect back before switching to discoverable mode. Default is 5.

5.19 **Current_ConnectBack_MAC_Address**

This string property returns the mac address of the last successfully connected Bluetooth device. Auto Connect feature will use this address while trying to auto connect.

5.20 **connectbackDeviceType**

This property returns the type of the last successfully connected Bluetooth device. Possible values are:

- Apple (iOS)
- SPP (Android or windows)
- NA (not applicable or unknown)

5.21 **IsSensorPowered**

This property returns the current status of the finger print sensor. It's true if sensor is in active state and ready to capture scans else false if sensor is in deep sleep state. You can use this property before going for finger capture.

5.22 **ConnectionPasswordEnabled**

This Boolean property returns the current state of Device access password. If device access password is enabled it's true otherwise will be false.

5.23 **stateOfCharge**

This float property returns the current battery level percentage.

5.24 **Voltage**

This integer property returns the current battery voltage level in millivolts.

5.25 **Current**

This integer property returns the battery current. It will be negative if battery is discharging and positive if battery is charging.

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

5.26 Temperature

This float property returns the temperature on the battery terminals in Celsius.

5.27 AbsoluteRemainingChargeValue

This integer property returns the absolute charge value remaining in battery.

5.28 powerSource

This property returns the current power source of the device. Possible values are:

- SMUFS_BATTERY = 1,
- SMUFS_USB_SRC = 2,
- SMUFS_EXTPWR = 3,
- SMUFS_PWR_UNKNOWN = 4

5.29 batteryState

This property returns the current state of the battery. Possible values are:

- BATTERY_CHARGING = 1,
- BATTERY_DISCHARGING = 2,
- BATTERY_LOW = 3,
- BATTERY_CRITICALLOW = 4,
- BATTERY_IDLE = 5,
- BATTERY_UNKNOWN = 6

5.30 IsCharging

This Boolean property returns true if battery is charging otherwise false.

| | | |
|-------------------------------------|------------------------|----------------------------|
| SMUFS-Android | Document ID: SMUFS-006 | Version: 0.3 |
| Application Program Interface (API) | | Version Date: Feb, 19,2015 |

6 Tutorials

This section will describe the example application(s) that are provided as part of the API.

6.1 SmufsCaptureSample

This simple Fingerscan capture demo app features :

1. Demonstrates sleep and wakeup features of the SDK. When user presses home button or device goes to sleep then this demo app will try to put the Smufs device to sleep as well. BT connection will be dropped in sleep state. While restarting the app will restore the connection back automatically.
2. In application Settings you can set to save all the captured finger images automatically in <SDCARD>/<smufscapturesample>/<Captured Images>. Images will be saved in BMP,JPG and WSQ formats in respective folders under Captured Images.
3. In Device settings you can use Auto Connect feature of the SDK, this feature attempts to reconnect to the last connected device if somehow Bluetooth connection is dropped during normal operations.
4. On battery status page, sample will show all the battery details of the Smufs device battery.
5. On Security settings page, you can enable or disable the password access control for the Smufs device.
6. On finger scan options page you can start and stop the sensor power in Smufs device.
7. General page displays the device properties like firmware version, USB VID/PID, MAC address of the device, MAC details of last connected device etc.

For details please refer to “SmufsCaptureSample” provided in Source folder of SDK package.