Classified:	Secret	Version:	V1.3.2
Range:	Customer	Status:	Release

HC21 Android SDK API User Guide

File No:	LT_UG_1618	Category	RD
Prepared:	Sky	Date	12/18/20
Checked:	Seaton	Date	12/18/20
Approved:	Thomas	Date	12/18/20

LINKTOP TECHNOLOGY CO., LTD.

Revision History

Version	Date	Description
V1.0.0	2020.08.28	Initial version English version; support Android 10.
V1.1.0	2020.12.18	Redefine API Interface.
V1.2.0	2021.02.03	1. This update improves the audio filter processing, and calculates the approximate heart rate value based on the audio waveform. 2. This update adds and improves the reading of device battery level, volume adjustment, echo mode switching, device information reading, and device software version and hardware version reading. 3. This update adds and removes (modified) some APIs. For details, please refer to the marked: API added since SDK version 1.2.0. and API removed since SDK version 1.2.0.
V1.3.0	2021.04.02	1. Modified the parameters of API ***connect(..)***. 2. Some APIs in ***DeviceInfo*** are deprecated, and the new add API ***getSN();*** can be used to get the SN from the device. 3. New add DFU feature, New APIs: ***dfuStart(…); dfuPause(); dfuResum(); dfuAbort();*** ***registerDfuProgressCallback(..);*** ***unregisterDfuProgressCallback (..);*** For details, please refer to the marked: API deprecated since SDK version 1.3.0. and API added since SDK version 1.3.0.
V1.3.1	2021.04.20	1. Filtering algorithm for distinguishing heart sounds and lung sounds; 2. The name of the echo mode has been changed from the original "Bell mode" and "Diaphragm mode" to "Heart sound mode" and "Lung sound mode". For details, please refer to the marked: API deprecated since SDK version 1.3.1

V1.3.2	2021.06.02	1. Fix the issue that the Bluetooth connection is disconnected due to timeout and cannot be reconnected; 2. Fix the issue of probabilistic Bluetooth connection status errors.

Directory

1. Overview

1.1Development Environment

Please use Android Studio to develop the project. The SDK is released in the form of an aar library. If you really need to use the jar library, please change aar to zip and unzip the file, extract the jar library inside and use it.

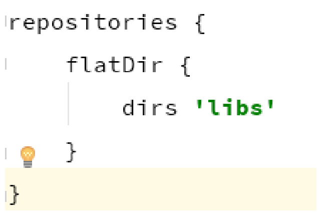
1.2 How to integrate

1.2.1 Library aar Configuration

1) build.gradle configuration:

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

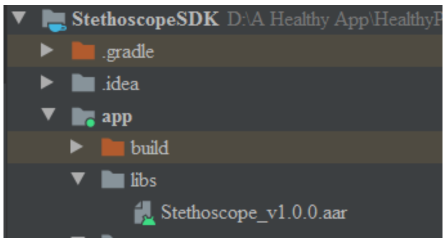
As follows:



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

2) Only integrate the Bluetooth communication module of the thermometer or health monitor, and do not use our server for data synchronization. Proceed as follows:

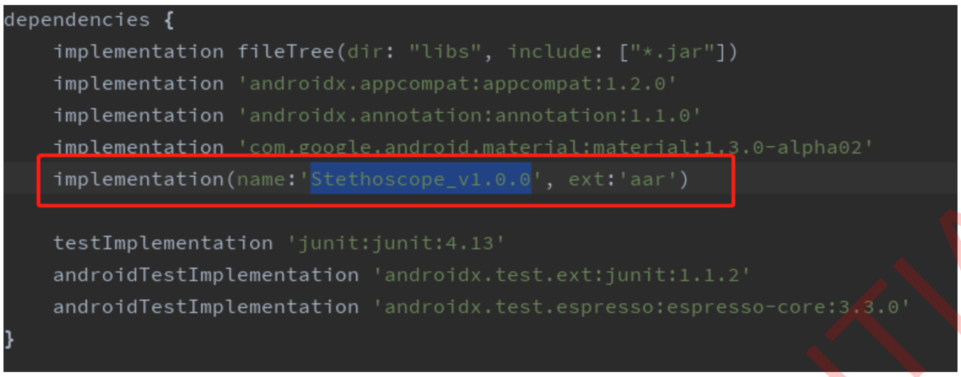
- Import Stethoscope_vx.x.x.aar in the \aar folder into the project \app\libs directory:



```
▼ StethoscopeSDK D:\A Healthy App\HealthyP
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▶ build
    ▼ libs
        Stethoscope_v1.0.0.aar
```

- Configure in the dependencies of build.gradle:

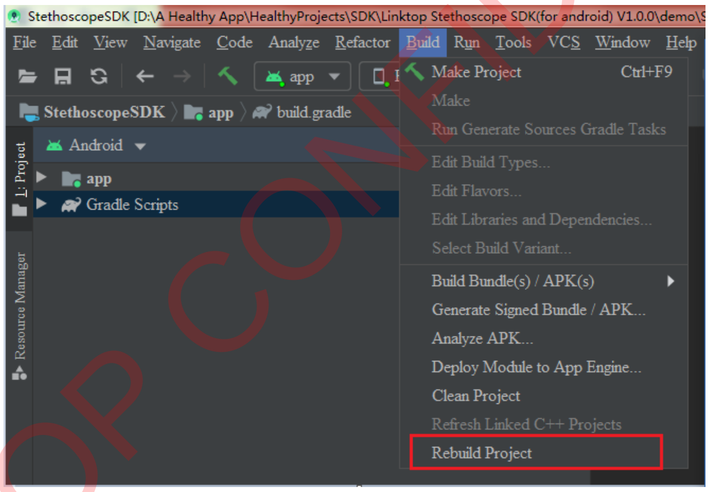
implementation(name:'Stethoscope_vx.x.x', ext:'aar');



```
dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.annotation:annotation:1.1.0'
    implementation 'com.google.android.material:material:1.3.0-alpha02'
    implementation(name:'Stethoscope_v1.0.0', ext:'aar')

    testImplementation 'junit:junit:4.13'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

- Click Build-Rebuild Project on the Android Studio menu bar to recompile the project:


StethoscopeSDK [D:\A Healthy App\HealthyProjects\SDK\Linktop Stethoscope SDK(for android) V1.0.0\demo\S
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app
StethoscopeSDK app build.gradle
Android
app
Gradle Scripts
1: Project
Resource Manager
Make Project Ctrl+F9
Make
Run Generate Sources Gradle Tasks
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Select Build Variant...
Build Bundle(s) / APK(s)
Generate Signed Bundle / APK...
Analyze APK...
Deploy Module to App Engine...
Clean Project
Refresh Linked C++ Projects
Rebuild Project

1.2.2 Dependent Library Configuration

Configure third-party dependency libraries in the dependencies of build.gradle, as follow:

```
implementation 'androidx.annotation: annotation:x.x.x'
```

1.2.3 AndroidManifest.xml Configuration

The permissions required by the SDK and the Application components used in the SDK have been configured in the .aar library. Developers only need to configure other permissions other than those required by the SDK. If it is suitable for the development of Android 6.0 or higher system, make the permission application that needs to dynamically apply for permission according to the following permission list:

Permission List of aar Library: Stethoscope_vx.x.x.aar

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

2. API

2.1 Fucntion

SDK connect to the device through BLE. You can easily communicate with the device by SDK , the equipment will provide auscultation data and other services. You can use this set of SDK to develop cardiac health monitoring applications based on Android system mobile devices. This SDK only provides cardiopulmonary sound and heart rate calculation for user development and testing.

2.2 Class *StethoscopeTool*

The StethoscopeTool class is mainly used to implement Bluetooth communication, record audio or play audio:

```
static void init(@NonNull Context context);
```

Initialize the SDK.

```
static void unInit();
```

Uninitialize the SDK.

```
static StethoscopeTool get()
```

Gets the singleton object of the stethoscopetool.

```
int getBleState();
```

Get the current BLE GATT status. There are three states at present:

Constants.BLE_GATT_DISCONNECTED;

Constants.BLE_GATT_CONNECTING;

Constants.BLE_GATT_CONNECTED.

~~void connect(@NonNull BleDevice bleDevice);~~

```
API deprecated since SDK version 1.3.0, use new:
void connect(@NonNull BluetoothDevice device);
```

Create BLE GATT connection with the device.

```
void disconnect();
```

Disconnect the BLE GATT from the device.

```
void startScan(OnBleDeviceScanListener listener);
```

Start scanning Bluetooth devices, the scan results will be called back with OnBleDeviceScanListener.

```
void stopScan()
```

Stop scanning Bluetooth devices.

```
void setEchoModeSwitch(@Constants.EchoMode int echoMode)
```

Setting auscultation mode mainly includes ~~Constants.ECHO_MODE_BELL (Bell mode)~~

and ~~Constants.ECHO_MODE_DIAPHRAGM (Diaphragm mode)~~.

```
API deprecated since SDK version 1.3.1, use new:
```

Constants.ECHO_MODE_HS (Heart sounds mode)

Constants.ECHO_MODE_LS (Lung sounds mode)

```
void startRecordAudio()
```

Start recording audio.

```
void stopRecordAudio()
```

Stop recording audio.

```
boolean isRecording()
```

Determine whether audio is being recorded. True is recording audio, otherwise false.

```
void readBatteryInfo()
API removed since SDK version 1.2.0 .
```

~~Read battery information.~~

```
void setBleWorkListener(OnBleWorkListener listener);
```

Set OnBleWorkListener callback interface.

```
void setOnStethoscopeDataListener(OnStethoscopeDataListener
listener);
```

Set stethoscope data callback interface.

```
public void adjustVolume(@Constants.Direction int direction);
API added since SDK version 1.2.0 .
```

Adjust the volume of the headset output of the device.

@param direction:

Constants.ADJUST_VOLUME_RAISE;

Constants.ADJUST_VOLUME_LOWER.

```
void readDeviceInfo()
```

Read device information.

```
void readDeviceVersion(@Constants.VersionType int type);
API added since SDK version 1.2.0 .
```

Read device version information, software version or hardware version.

@param type:

Constants.VERSION_TYPE_HARDWARE;

Constants.VERSION_TYPE_SOFTWARE;

```
public void setOnDeviceInfoListener(OnDeviceInfoListener listener);
API added since SDK version 1.2.0 .
```

Set OnDeviceInfoListener callback interface.

```
public void dfuStart(@NonNull BluetoothDevice device,
              @Nullable Uri fileStreamUri,
              @Nullable String filePath,
              @NonNull final Class<? extends DfuBaseService>
service);
API added since SDK version 1.3.0 .
```

Start the DFU process.

@param device: the target device object.

@param fileStreamUri: the URI of the upgrade file (.zip).

@param filePath: the path of the upgrade file (.zip).

@param service: a Service class derived from the abstract class BaseDfuService.

```
void dfuPause();
API added since SDK version 1.3.0
```

Pause the DFU process.

```
void dfuAbort();
API added since SDK version 1.3.0
```

Abort the DFU process

```
void registerDfuProgressCallback(OnDfuProgressCallbackAdapter
callback);
API added since SDK version 1.3.0
```

Register DFU progress callback.

```
void unregisterDfuProgressCallback(DfuProgressCallbackAdapter
callback);
API added since SDK version 1.3.0
```

Unregister DFU progress callback.

2.3 Interface *OnBleWorkListener*

```
void onBluetoothStateChanged(int state);
```

Call back when Bluetooth status changes.

@param state:

android.bluetooth.BluetoothAdapter.STATE_OFF;

android.bluetooth.BluetoothAdapter.STATE_TURNING_ON;

android.bluetooth.BluetoothAdapter.STATE_ON;

android.bluetooth.BluetoothAdapter.STATE_TURNING_OFF.

```
void onBleGattStateChanged(int state, BluetoothDevice device);
```

Callback when BLE GATT status changes.

@param state：

Constants.BLE_GATT_DISCONNECTED;

Constants.BLE_GATT_CONNECTING;

Constants.BLE_GATT_CONNECTED;

@param device:

When state = Constants.BLE_GATT_CONNECTED is not null. It is the currently connected device object.

2.4 Interface *OnBleDeviceScanListener*

```
void onBleDeviceList(List<BleDevice> list);
```

Callback scanned Bluetooth stethoscope device list.

2.5 Interface *OnSethoscopeDataListener*

```
~~void onAudioBuffer(short[] buffer)~~
API removed since SDK version 1.2.0 .
```

~~Call back audio buffer array.~~

```
void onFilteredAudioData(short[] data);
API added since SDK version 1.2.0 .
```

Call back the filtered audio buffer array.

```
void onRawAudioData(short[] data);
```

```
API added since SDK version 1.2.0 .
```

Call back the raw audio buffer array.

```
void onSynchronizingDeviceData(boolean inProgress);
API added since SDK version 1.2.0 .
```

After the device is connected, some data will be synchronized, including the device power level, device volume level and current diagnostic mode.

@param inProgress:

true: In the synchronization progress.

false: The synchronization is complete.

```
public void onBatteryLevelChanged(int level);
API added since SDK version 1.2.0 .
```

Callback when the battery level of the device changes.

@param level: Range 0~100.

```
public void onVolumeLevelChanged(int level);
API added since SDK version 1.2.0 .
```

Callback when the volume level of the device changes.

@param level: Range 0~10.

```
void onEchoModeSwitch(@Constants.EchoMode int echoMode)
API added since SDK version 1.2.0 .
```

Callback when the device's auscultation mode changes.

@param echoMode:

~~Constants.ECHO_MODE_BELL~~;

~~Constants.ECHO_MODE_DIAPHRAGM~~.

`API deprecated since SDK version 1.3.1, use new:`

Constants.ECHO_MODE_HS;

Constants.ECHO_MODE_LS.

```
void onResult(int heartRate);
```

The heart rate value of the callback. Calculated with audio data.

```
void onException(int exception)
API added since SDK version 1.2.0 .
```

Callback when some exception occur in the SDK.

@param:

Constants.EXCEPTION_VOLUME_LEVEL_MAX;

Constants.EXCEPTION_VOLUME_LEVEL_MIN;

Constants.EXCEPTION_SYNC_DATA_TIMEOUT;

2.6 Interface *OnDeviceInfoLstener*

`API added since SDK version 1.2.0`

```
void onDeviceInfo(@NonNull DeviceInfo deviceInfo)
```

Callback device production information.

@param deviceInfo: An object containing production information.

```
onDeviceVersion(@Constants.VersionType int type, String version)
```

Callback the device version information read.

@param type: version type

@param version: version name.

2.7 Class *DeviceInfo*

`API added since SDK version 1.2.0`

This class contains the factory's production information read from the device.

```
~~String getYear();~~
API deprecated since SDK version 1.3.0
```

~~The production year of the device.~~

```
~~String getMonth();~~
API deprecated since SDK version 1.3.0
```

~~The production month of the device.~~

```
~~String getFactory();~~
API deprecated since SDK version 1.3.0
```

~~The code of the factory producing this device.~~

```
~~String getDeviceType();~~
API deprecated since SDK version 1.3.0
```

~~Device type.~~

```
String getDeviceId();
```

Device ID.

```
~~String getDeviceKey();~~
API deprecated since SDK version 1.3.0
```

~~Device key.~~

```
String getAKey();
```

Device aKey.

```
~~String getReserved();~~
API deprecated since SDK version 1.3.0
```

~~Reserved fields, temporarily useless.~~

```
String getSn();
API added since SDK version 1.3.0 .
```

The SN of the device. Use this string information to replace all the above outdated string information.

2.8 Class *Constants*

Define the pattern of the stethoscope:

```
~~public final static int ECHO_MODE_BELL;~~
API deprecated since SDK version 1.3.1, use new:
public final static int ECHO_MODE_HS;
```

Heart sounds mode.

```
~~public final static int ECHO_MODE_DIAPHRAGM ;~~
API deprecated since SDK version 1.3.1, use new:
public final static int ECHO_MODE_LS;
```

Lung sounds mode.

Define the BLE GATT state:

```
public final static int BLE_GATT_DISCONNECTED;
```

GATT disconnected.

```
public final static int BLE_GATT_CONNECTING;
```

GATT connecting.

```
public final static int BLE_GATT_CONNECTED;
```

GATT connected.

Define the direction of volume adjustment:

```
API added since SDK version 1.2.0
public static final int ADJUST_VOLUME_LOWER;
```

Decrease the device volume.

```
public static final int ADJUST_VOLUME_RAISE;
```

Increase the device volume.

Define the type of reading device version:

```
API added since SDK version 1.2.0
public static final int VERSION_TYPE_HARDWARE;
```

Hardware version type

```
public static final int VERSION_TYPE_SOFTWARE;
```

Software version type

Some exceptions:

```
API added since SDK version 1.2.0
public static final int EXCEPTION_VOLUME_LEVEL_MAX;。
```

When the volume is increased to the maximum level, continuing to increase the volume will trigger the exception.

```
public static final int EXCEPTION_VOLUME_LEVEL_MIN;
```

When the volume is decreased to the minimum level, continuing to decrease the volume will trigger the exception.

```
public static final int EXCEPTION_SYCN_DATA_TIMEOUT;
```

When the BLE is connected to the device to synchronize data, if the process cannot be completed for a long time, the exception will be triggered.

2.9 Abstract class *BaseDfuService*

BaseDfuService is an abstract class that integrates all the processes of DFU, and it extends from Service. You need to create a new Service class declared in AndroidManifest.xml to extends from it.

```
@Nullable
protected abstract Class<? extends Activity> getNotificationTarget
();
```

This method must return the activity class that will be used to create the pending intent used as a content intent in the notification showing the upload progress or service foregro und state.

@return: The target activity class.

```
protected void updateErrorNotification (NotificationCompat.Builder
builder);
```

This method allows you to update the notification showing an error.

@param builder: Error notification builder.

```
protected void updateForegroundNotification
(NotificationCompat.Builder builder);
```

This method allows you to update the notification that will be shown the service goes to the foreground state.

@param builder: Foreground notification builder.

```
protected void updateProgressNotification
(NotificationCompat.Builder builder, int progress);
```

This method allows you to update the notification showing the upload progress.

@param builder: Notification builder.

@param progress: Upload progress.

2.10 OnDfuProgressCallbackAdapter

API added since SDK version 1.3.0

This is the callback class of the DFU process。

```
void onDeviceConnecting(String deviceAddress);
```

Method called when the DFU service started connecting the DFU target.

@param deviceAddress: Bluetooth address of the target device.

```
void onDeviceDisconnecting(String deviceAddress);
```

Method called when the service disconnected from the device. The device has been reset.

@param deviceAddress: Bluetooth address of the target device.

```
void onDfuProcessStarting(String deviceAddress);
```

Method called when the DFU process is starting.

@param deviceAddress: Bluetooth address of the target device.

```
void onEnablingDfuMode(String deviceAddress);
```

Method called when the service discovered that the DFU target is in the application mode and must be switched to DFU mode.

@param deviceAddress: Bluetooth address of the target device.

```
void onFirmwareValidating(String deviceAddress);
```

Method called when the new firmware is being validated on the target device.

@param deviceAddress: Bluetooth address of the target device.

```
void onDfuCompleted(String deviceAddress);
```

Method called when the DFU process succeeded.

@param deviceAddress: Bluetooth address of the target device.

```
void onDfuAborted(String deviceAddress);
```

Method called when the DFU process has been aborted.

@param deviceAddress: Bluetooth address of the target device.

```
void onProgressChanged(String deviceAddress, int percent, float
speed, float avgSpeed,
   int currentPart, int partsTotal);
```

Method called during uploading the firmware. It will not be called twice with the same value of percent, however, in case of small firmware files, some values may be omitted.

@param deviceAddress: Bluetooth address of the target device.

@param percent: The current status of upload (0~99).

@param speed: The current speed in bytes per millisecond.

@param avgSpeed: The average speed in bytes per millisecond.

@param currentPart: The number pf part being sent. In case the ZIP file contains a Soft Device and/or a Bootloader together with the application the SD+BL are sent as part 1, then the service starts again and send the application as part 2.

@param partsTotal: Total number of parts

```
void onError(String deviceAddress, int error, float errorType,
String message);
```

Method called when an error occur.

@param deviceAddress: Bluetooth address of the target device.

@param error: The error number.

@param errorType: The error type, one of

{*@link DfuBaseService#ERROR_TYPE_COMMUNICATION_STATE*},

{*@link DfuBaseService#ERROR_TYPE_COMMUNICATION*},

{*@link DfuBaseService#ERROR_TYPE_DFU_REMOUTE*},

{*@link DfuBaseService#ERROR_TYPE_OTHER*}.

@param message: The error message.

2.11 DFU integration instructions

Device Firmware Upgrade (DFU), the firmware of the device can be upgraded by sending a firmware package (zip) to the device through the Bluetooth of the mobile phone, in order to achieve the iterative purpose of the device firmware version.

1. Need to increase the configuration of three dependent packages

```
implementation 'androidx.core:core:x.x.x'
implementation
'androidx.localbroadcastmanager:localbroadcastmanager:x.x.x'
implementation 'com.google.core.gson:gson:x.x.x'
```

2. Create a new Service to extend ***DfuBaseService***.

All the processes of DFU have been integrated in ***DfuBaseService***, which is an abstract class and is defined as a front-end Service, so you must declare the newly created Service in ***AndroidManifest.xml***. And the permissions required by the foreground Service

```
<uses-permission
android:name="android.permission.FOREGROUND_SERVICE" />
```

has been declared in the AndroidManifest.xml of the SDK ***aar*** package, so you don't need to repeat the declaration. After Service inherits ***DfuBaseService***, the main function is to provide some method rewriting so that you can customize your own UI. At present, the notification styles used in the main ***DfuBaseService*** process will be further explained in the following process.

3. Create a new target Activity as the target activity of the resident notification of the foreground Service, rewrite the abstract method ***getNotificationTarget()*** in the Service, and return the Activity.

E.g:

```
@Override
protected Class<? extends Activity> getNotificationTarget() {
    return NotificationActivity.class;
}
```

As a target activity, ***NotificationActivity*** will be returned instead of ***MainActivity***. This is because when the user presses the notification, the notification must create a new task:

```
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

Using ***NotificationActivity***, we can check whether the new activity is a target activity (which means that no other activity has been opened before) or another activity has already been opened. In the latter case, ***NotificationActivity*** will be closed. The system will restore the previous activity. However, if the application is closed during the upload and the user clicks on the notification, the ***NotificationActivity*** will start as a target activity. It will create and start ***MainActivity*** and terminate itself.

When the application is closed or opened, you can use this method to resume target activity. It can also be used to re-create the activity history (see ***NotificationActivity*** of the Demo project for details).