

Instructions on SDK for Htek Android Phones

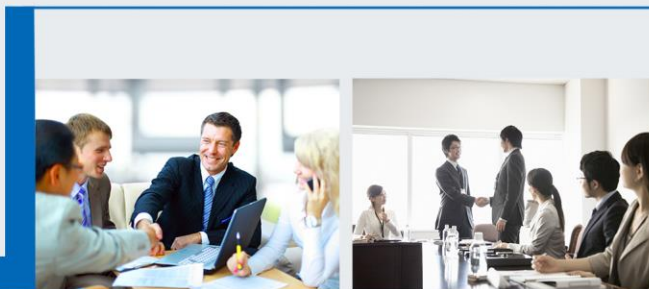


Table of Contents

1 SDK Introduction.....	3
2 Key Events.....	3
2.1 Dispatching Keys When APP is Running.....	3
2.2 Special Keys.....	4
2.2.1 Monitor the change of handset off-hook state	4
3 Power LED Indicator.....	5
3.1 Making the LED Indicator On	5
3.2 Making the LED Indicator Flash	5
3.3 Making the LED Indicator Off	6
4 Voice Channel.....	6
4.1 Speaker Mode	6
4.2 Headset Mode	6
4.3 Handset Mode	7
4.4 Get Speaker status.....	7
5 Device Control	7
5.1 Control the Device Restart	7

1 SDK Introduction

This SDK provides some interfaces to developers. It helps the developers to quickly access the Htek Android phone platform.

2 Key Events

2.1 Dispatching Keys When APP is Running

You can intercept all key events when they are dispatching. Or do some operations when the user presses or releases the key

Scenario: Listen the Mute key when in third-party APP screen.

```
@Override
public boolean dispatchKeyEvent(KeyEvent event) {
    if(event.getKeyCode() == KeyEvent.KEYCODE_MUTE){
        // TODO
        return true;
    }
    return super.dispatchKeyEvent(event);
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    If(keyCode == KeyEvent.KEYCODE_MUTE){
        // TODO
        return true;
    }
    return super.onKeyDown(keyCode, event);
}

@Override
public boolean onKeyUp(int keyCode, KeyEvent event) {
    If(keyCode == KeyEvent.KEYCODE_MUTE){
        // TODO
        return true;
    }
    return super.onKeyUp(keyCode, event);
}
```

For V20:

There are some keys on its side, and the value of keyevent is as follows. You can monitor each key by replacing the key value in the above scenario;

```
Hold key:
KEYEVENT_HOLD = KeyEvent.KEYCODE_F3;
Forward key:
KEYEVENT_FORWARD = KeyEvent.KEYCODE_F2;
Redial key:
KEYEVENT_REDIAL = KeyEvent.KEYCODE_F1;
Headset key:
KEYEVENT_HEADSET = KeyEvent.KEYCODE_F5;
Mute key:
KEYEVENT_MUTE = KeyEvent.KEYCODE_MUTE;
Speaker key:
KEYEVENT_SPEAKER = KeyEvent.KEYCODE_F4;
Message key:
KEYEVENT_MESSAGE = KeyEvent.KEYCODE_ENVELOPE;
```

2.2 Special Keys

2.2.1 Monitor the change of handset off-hook state

Step1: Configure the intentFilter

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(CallConstant.ACTION_HANDSET_PLUGIN);
```

Step2: Init BroadcastReceiver

```
private final BroadcastReceiver mBroadcastReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent == null) {
            return;
        }
        if (intent.getAction().equals("android.intent.action.HANDLE_PLUG")) {
            if (intent.hasExtra("state")) {
                int state = intent.getIntExtra("state", -1);
                // TODO
            }
        }
    }
}
```

```
    }  
  }  
};
```

Step3: register Broadcast Receiver

```
registerReceiver(mBroadcastReceiver, intentFilter);
```

3 Power LED Indicator

V20 supports red and green LED, and V22 supports red, green and blue LED.

3.1 Making the LED Indicator On

Scenario: Make the Power LED indicator on when the APP is ready.

You can set color of Power LED by:

```
PowerLed.getInstance(getApplicationContext()).setPowerLedState(PowerLed.STATE.RED_ON);
```

Or use turn2State, function turn2State will stop the flash of LED at the same time:

```
PowerLed.getInstance(getApplicationContext()).turn2State(PowerLed.STATE.RED_ON);
```

3.2 Making the LED Indicator Flash

Scenario: Make the Power LED indicator flash when there is an incoming call arrived on the APP.

The three parameters are the two states of the light flashing and the flashing interval in milliseconds

```
PowerLed.getInstance(getApplicationContext()).setFlashState(PowerLed.STATE.OFF, PowerLed.STATE.GREEN_ON, 100);  
PowerLed.getInstance(getApplicationContext()).startFlash();
```

3.3 Making the LED Indicator Off

Scenario: Make the Power LED indicator off when rejecting the incoming Call.

You can stop flash of Power LED by:

```
PowerLed.getInstance(getApplicationContext()).turn2State(STATE.OFF);
```

Or use function turnOff():

```
PowerLed.getInstance(getApplicationContext()).turnOff();
```

4 Voice Channel

You can use the API interface to switch the voice channel, get the currently used voice channel and implement the corresponding process according to the voice channel.

Step1: Add Audio Permission to AndroidManifest.xml

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

Step2: switch voice channel

4.1 Speaker Mode

Scenario: Switch the voice channel to speaker mode when the third-party APP is running.

```
VoiceChannelManager.getInstance(getApplicationContext()).setVoiceChannel(VoiceChannel.MODE_ HANDFREE);
```

4.2 Headset Mode

Scenario: Switch the voice channel to headset mode when the third-party APP is running.

```
VoiceChannelManager.getInstance(getApplicationContext()).setVoiceChannel(VoiceChannel.MODE_
HEADSET);
```

4.3 Handset Mode

Scenario: Switch the voice channel to handset mode when the third-party APP is running.

```
VoiceChannelManager.getInstance(getApplicationContext()).setVoiceChannel(VoiceChannel.MODE_
HANDSET);
```

4.4 Get Speaker status

The on-off state of the speaker can be obtained through the return value of function isSpeakerphoneOn();

```
VoiceChannelManager.getInstance(getApplicationContext()).isSpeakerphoneOn();
```

5 Device Control

5.1 Control the Device Restart

(The reboot interface has not been opened yet. Please contact technical support if necessary)

```
Device.getInstance(getApplicationContext()).reboot();
```