```
//
//  SoundWaveWifiSetting.java   WiFiMatching.java
//  SoDemon
//
------------------------------------------------------------
```

SoundWaveWifiSetting & WiFiMatching

说明：
无线网络设置，是指把摄像机网络设置连接到某一个无线路由器。
无线网络设置有四种方式：声音设置无线网络，smartlink设置无线网络和单纯的无线设置，web_app。
其中无线设置添加有两种方式：声音设置无线网络，smartlink设置无线。这两种方式可以单独使用，也可以同
时使用。但同时使用时，一定要注意。当一种方式成功时，一定要记得把另外一种设置方式停止。即smartlinK
设置先成功时，要记得把sound停止。当sound设置先成功时，要记得把smartlink停止。

#1.声音设置无线 <无线设置 / 停止 / 无线设置结果 / 结果信息>
1.声音设置无线(声波设置无线，APK文件需签名打包才行，且签名文件右reecam提供)

1）声音设置的初始化和反初始化
说明：使用声音设置无线，需先进行初始化，设置完成后必须反初始化，初始化和反初始化需成对使用
函数：public void init(Context context); //声波初始化
      public void deinit(); //声波反初始化

2）开始声音设置无线
说明：使用以下函数可以通过声音设置无线网络也可以监控设置的过程。
函数：public boolean start(String ssid, String psk, sound_wave_wifi_setting_listener listener)

```
/*
 * 参数如下：
 * ssid:wifi的ssid
 * psk:wifi的密码
 * listener:接口，当前对象需实现sound_wave_wifi_setting_listener接口
 * 返回：
 * 返回一个BOOL值，当返回true时，表示函数执行成功，否则执行失败。
 */

/*
 *这是接口sound_wave_wifi_setting_listener的回调函数
 *开始声音设时，添加了此接口
 *只要声音设置无线状态有改变时候就会调用此函数
 */
```

回调函数：public void on_sound_wave_wifi_setting_state_changed();

2）停止声音设置无线
说明：使用此函数，可以强制停止声音无线设置
函数：public void stop();

3）声音设置无线信息可从回调函数中获取：

```
/*
 *声音设置无线结果，共有两种结果，详细如下：
 *声音设置结果通过public SOUND_WAVE_WIFI_SETTING_RESULT result()获取
 */
public static enum SOUND_WAVE_WIFI_SETTING_RESULT {
        SUCCEED, //设置成功
        TIMEOUT  //设置超时，需重新设置
```

```
    }

/*
 *声音设置无线状态，共有三种状态，详细如下：
 *状态通过public SOUND_WAVE_WIFI_SETTING_STATE state()
 */
public static enum SOUND_WAVE_WIFI_SETTING_STATE {
        IDLE,           //状态为空时，可查询当前设置结果，进行不同操作
        SEND_SETTING,   //正在发送Wifi设置信息
        WAIT_CONFIRM    //等待验证，即等待设备响应
    };


/*设置成功后，查询id是否在局域网内
 *因为id太长，摄像机传不过来，所以把id编码分成name_of_camera_id和serial_of_camera_id来传
 *特征码1和特征码2说明：
 *  1.设置wifi成功的机器，可在局域网中通过IPCamMgr.get_discovered_cameras_list()方法搜索到；
 *
 2.得到该camera的id后，可通过Tools.get_name_of_camera_id(id)的方法得到一个特征码m1，通过Tools.
 get_serial_of_camera_id(id)的方法得到一个特征码m2；
 *
 3.第二步中得到的特征码m1和该编码生成的特征码1进行比对，并且m2和该编码生成的特征码2相等则此ID
 即为当前声波设置wifi成功的机器；
 */
public int name_of_camera_id();
public int serial_of_camera_id() ;

4）示例：
/*
 第一步：创建一个SoundWaveWifiSetting对象,初始化声波设置
 */
private SoundWaveWifiSetting m_sound_wave_wifi_setting = new SoundWaveWifiSetting();
m_sound_wave_wifi_setting.init(m_parent);
/*
 *第二步：发送声音 实现接口sound_wave_wifi_setting_listener
 */
 boolean  m_sound_wifi = m_sound_wave_wifi_setting.start(m_ap.ssid, m_ap.key,
 AddCamerabySoundDialog.this);
   if (! m_sound_wifi){
        Log.e("SoDemo","SEND_WIFI_INFO_FAIL");
    }

/*
 *监测 声音设置无线的过程
 */
public void on_sound_wave_wifi_setting_state_changed() {
    SOUND_WAVE_WIFI_SETTING_STATE state = m_sound_wave_wifi_setting.state();
    SOUND_WAVE_WIFI_SETTING_RESULT result = m_sound_wave_wifi_setting.result();
    Log.e("SoDemo","SOUND_WAVE_WIFI_SETTING_STATE = " + state +
    "SOUND_WAVE_WIFI_SETTING_RESULT = " + result);
    if (state == SOUND_WAVE_WIFI_SETTING_STATE.SEND_SETTING) {
            Log.e("SoDemo","Transmitting wireless setting now");//正在传送无线设置 ...;
    } else if (state == SOUND_WAVE_WIFI_SETTING_STATE.WAIT_CONFIRM) {
            Log.e("SoDemo","Wait device response");//正在等待设备响应 ...;
    } else if (state == SOUND_WAVE_WIFI_SETTING_STATE.IDLE) {
        if (result == SOUND_WAVE_WIFI_SETTING_RESULT_SUCCEED){
            m_name_of_camera_id = m_sound_wave_wifi_setting.name_of_camera_id();
```

```
            m_serial_of_camera_id = m_sound_wave_wifi_setting.serial_of_camera_id();
            Log.e("SoDemo","Search camera in LAN");//设置成功可在局域网开始搜索此机器
            discover_camera();
        }
    }
}

/*
 *第三步：设置成功后，查询id是否在局域网内,来获取id
 */
public void discover_camera(){
    LinkedHashMap<String, DISCOVERED_CAMERA_INFO> cameras = IPCamMgr.
    get_discovered_cameras_list();
    String camera_id = null;
    int name_of_camera_id;
    int serial_of_camera_id;

    for (String id : cameras.keySet()) {
        /*对局域网的摄像机id进行编码*/
        name_of_camera_id = Tools.get_name_of_camera_id(id);
        serial_of_camera_id = Tools.get_serial_of_camera_id(id);
        /*查id 是否在局域网内*/
        if (name_of_camera_id == m_name_of_camera_id &&
            serial_of_camera_id == m_serial_of_camera_id) {
            camera_id = id;
            Log.e("SoDemo","Camera id is" + id);
            break;
        }
    }
/*
 *不需要了，强制取消SoundWaveWifisetting设置。设置成功了，不需要做停止的的操作
 */
    m_sound_wave_wifi_setting.stop();
    m_sound_wave_wifi_setting.deinit();
```

#2.smartlink设置无线网络 <设置 / 停止 / 成功回调>

2.smartlink设置无线网络

1） smartlink设置

说明：使用以下函数可以通过smartlink设置无线网络

```
/*
 *参数如下：
 *ssid:wifi的ssid
 *psk:wifi的密码
 *listener:wifi_matching_listener接口
 * 返回：
 * 返回一个boolean值，当返回true时，表示函数执行成功，否则执行失败。
 */
```

函数：public static boolean start(String ssid, String psk, wifi_matching_listener listener)

```
/*
 *这是接口wifi_matching_listener的回调函数
 *开始smartlink设置时时，实现此接口
 *当smartlink设置无线成功后就会调用此函数且返回成功摄像机的id,其他时候不会掉用此函数
 */
```

回调函数：

```
public void on_wifi_matching_succeed(String camera_id);
```

2）停止smartlink设置无线

说明：此函数可以强制停止smartlink设置无线

函数：

```
public static void stop();
```

3）示例：

```
/*
 *第一步：创建一个WiFiMatching对象
 */
    private WiFiMatching m_smart_link_wifi_setting = new WiFiMatching();

/*
 *第二步：发送smartlink 实现接口wifi_matching_listener
 */
    boolean  m_smart_link = m_smart_link_wifi_setting.start(m_ap.ssid, m_ap.key,
    AddCamerabySmartDialog.this);;
    if (! m_smart_link){
       Log.e("SoDemo","Smart Link start OK!");
    }

/*
 *wifi_matching_listener回调函数
 */
public void on_wifi_matching_succeed(String camera_id){
    if (camera_id != "") {
        Log.e("SeDemo","smartlink设置成功, camera_id= " + camera_id);
        }
}

/*
 *不需要了，强制取消smartlink设置，设置成功了，需要做停止的的操作
 */
    m_smart_link_wifi_setting.stop();
```