云对讲客户端 Android SDK 开发指南

一、快速体验

在 Demo 程序中，演示了咚咚云对讲平台提供的账号管理、客户端与云对讲设备之间可视对讲的功能。

1.1 环境搭建

Windows 系统。开发工具：Android Studio

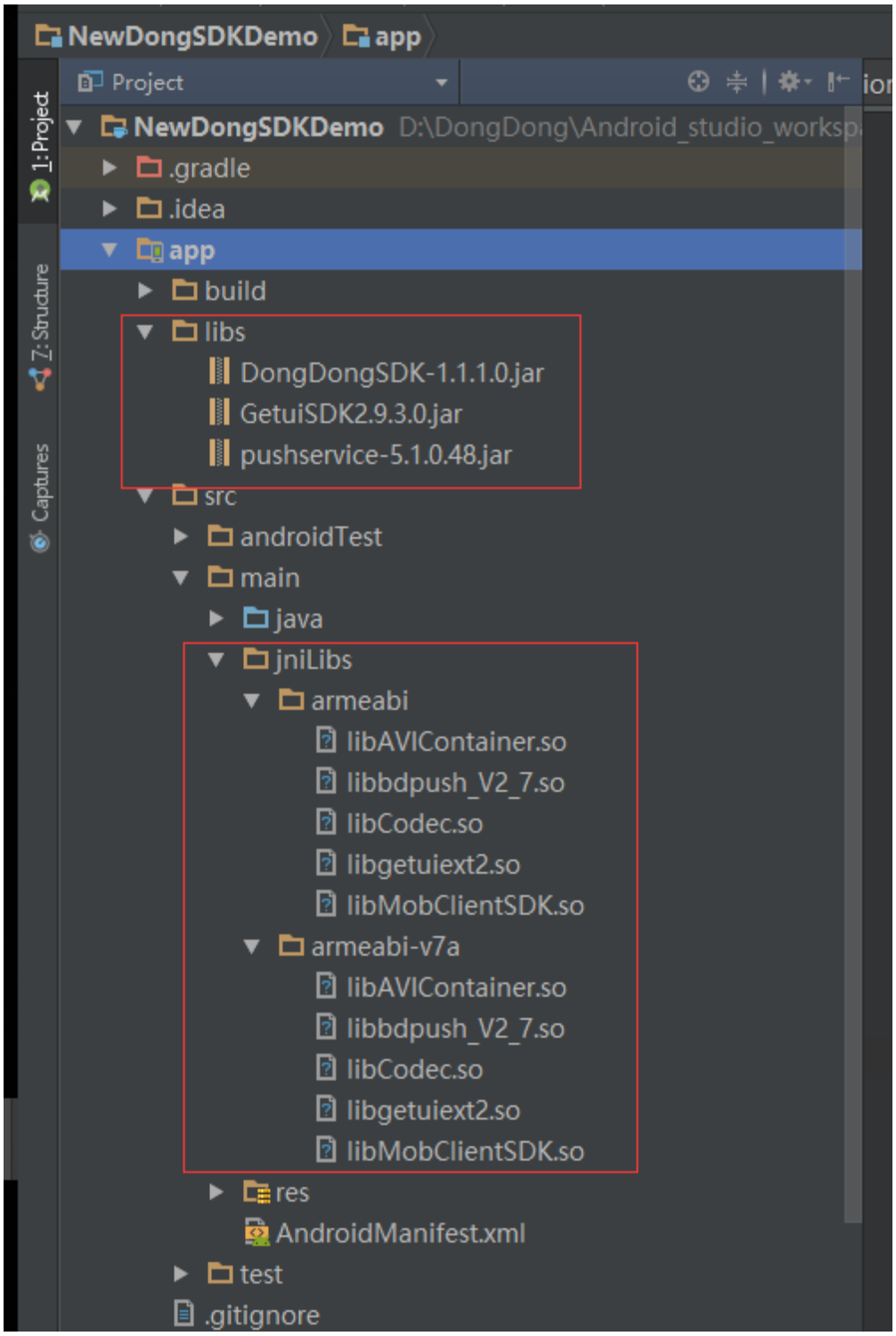
1.2Demo 介绍

DongSDKDemo 功能介绍，Demo 演示了 DongSDK 的 API 接口调用，主要实现的功能：

- 设备音视频：查看设备音频、视频及语音对讲
- 账号及设备管理：注册及登录账号，可授权设备
- 局域网：通过局域网获取设备进行音视频连接

1.3 导入 DEMO 工程(暂未支持 Maven 自动导入)

Android Studio 导入 demo，导入过后程序还是不能运行的，我们还需要 DongSDK-xxx.jar、GetuiSDK-xxx.jar、pushservice-xxx.jar 文件文件复制到 app 模块目录下的 libs文件夹中，将 so 文件夹子文件复制到 app/src/main/jniLibs 模块中。如下图：



打开 app/build.gradle，在 dependencies 中添加相关 jar 包的引用：

```
dependencies {
    compile files('libs/DongDongSDK.jar')
    compile files('libs/GetuiSDK2.9.0.0.jar')
    compile files('libs/pushservice-5.1.0.48.jar')
}
```

注：导入需要的 cpu 架构的 so 库即可，如果 app/build.gradle文件中的 android{}中添加了 jnilib 引用，那么将 so 文件复制到 app/libs 模块中。代码如下：

```
sourceSets {
    main {
        jniLibs.srcDirs = ['libs']
    }
}
```

1.4 配置 SDK 参数

如果只是演示 Demo 功能用，此步骤可以跳过，如果正式集成 SDK 到应用时，那此步骤是必须的，这里会告诉你如何配置相应参数。

1.4.1 迁移 Demo 中的参数

将 AndroidManifest 文件中的权限、咚咚生成的 AppId、推送信息全部移到需要集成 SDK 应用的 AndroidManifest 中，如下图:

```
<!--需要将以下权限拷贝到集成SDK的应用中去-->
<uses-permission android:name="baidu.push.permission.
    WRITE_PUSHINFOPROVIDER.${APP_PACKAGE_NAME_VALUE}" />
<permission
    android:name="getui.permission.
    GetuiService.${APP_PACKAGE_NAME_VALUE}"
    android:protectionLevel="normal" />
<!--需要将以上权限拷贝到集成SDK的应用中去-->
```

```
<!--
 ////////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////
 以下内容全部复制到需要集成SDK的应用的AndriodManifest.xml中///
 ////////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////
 -->
<!-- 配置DongSDK参数属性 -->
<meta-data
    android:name="DONG_APPID"
    android:value="${DONG_APPID_VALUE}" />
<!-- 个推SDK配置开始 -->
<!-- 配置的第三方参数属性 -->
<meta-data
    android:name="PUSH_APPID"
    android:value="${GT_PUSH_APPID_VALUE}" />
<meta-data
    android:name="PUSH_APPSECRET"
```

将箭头以下内容全部拷贝

1.4.2 修改 app/build.gradle 下的配置参数

替换 app/build.gradle 文件中 defaultConfig 信息，如下图:

```
//编译配置
defaultConfig {
    manifestPlaceholders = [
            BUILD_TIME_VALUE:new Date().format('yyyyMMdd'),
            DONG_APPID_VALUE :'Your dong appId',
            GT_PUSH_APPID_VALUE :'Getui appId',
            GT_PUSH_APPSECRET_VALUE:'Getui secret',
            GT_PUSH_APPKEY_VALUE :'Getui appKey',
            APP_PACKAGE_NAME_VALUE :'Your package name',
            BAIDU_API_KEY_VALUE   :'Baidu api key'
    ]
}
```

二、接口

2.1 云平台 SDK 初始化/结束

注:下面接口在 DongSDK.java 类中

2.1.1 初始化 SDK

```
/**
*  初始化 SDK                    在程序启动的时候需要首先初始化 sdk
*@param context                 程序运行上下文
* @return result                 0:成功,1:解析出错,2,AppId 不匹配
*/
public static int initDongSDK (Context context);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时
*  初始化 SDK                    在程序启动的时候需要首先初始化 sdk
* return result                 true:成功,false:失败
*/
public static boolean dongSdk_Init ();
```

2.1.2 结束 SDK

```
/**
*  结束 SDK                      在退出程序的时候需要调用该函数结束 SDK
*/
public static void finishDongSDK ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时
*  结束 SDK                      在退出程序的时候需要调用该函数结束 SDK
*/
public static void dongSdk_Finish ();
```

2.1.3 重复初始化 SDK

```
/**
* 重复初始化 SDK                重复初始化 sdk
* return result                 true:成功,false:失败
*/
public static boolean reInitDongSDK ()
```

2.1.4 初始化推送

```
/**
* 初始化推送                    建议最好放在 Activity 中初始化，只需调用一次
* @param context                程序运行上下文
* @param platform               1:个推, 2:百度,3:全部
*/
public static void initializePush (Context context, int platform);
```

2.2 账号管理(查询、注册、登录、注销)

注：下面未过时接口在 DongSDKProxy.java 类中， @Deprecated 标注过时接口在 DongdongAccount.java 类中，其中查询、注册功能必须先调用 DongSDKProxy. intDongRegister 方法，登录、注销必须先调用 DongSDKProxy. initDongAccount 方法。

2.2.1 初始化用户注册、查询功能实体

```
/**
* 初始化 DongRegister           建议全局只初始化一次
* @param callback               回调监听者
*/
public static void intDongRegister (DongRegisterCallback callback);
```

2.2.2 判断 DongRegister 的初始化状态

```
/**
* 初始化 DongRegister           判断是否实例化 DongRegister 时调用
* @return result                true:已经初始化完,false:未初始化
*/
```

```
public static boolean initCompleteDongRegister ();
```

2.2.3 注册回调监听者

```
/**
* 注册 DongRegister 功能的回调监听者    建议用在界面 onResume 方法中
*@param impl                    被注册的回调监听者
*/
public static void registerDongRegisterCallback (DongRegisterCallbackImp impl);
```

2.2.4 解绑回调监听者

```
/**
* 解绑 DongRegister 功能的回调监听者    建议用在界面 onPause 方法中
*@param impl                    被解绑的回调监听者
*/
public static void unRegisterDongRegisterCallback (DongRegisterCallbackImp impl);
```

2.2.5 查询账号是否存在

```
/**
* 查询账号                      查询账号是否存在
* @param username               用户名（手机客户端目前只针对手机号码和邮箱）
* return result                 0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestQueryUser (String userName)
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        查询账号是否存在
* @param username               用户名（手机客户端目前只针对手机号码和邮箱）
* return result                 0:指令发送成功，其它数字则为指令发送失败
*/
public int queryUser (String username);
```

2.2.6 发送短信获取验证码

```
/**
* 发送短信验证码（手机号码作为账号进行注册）
* @param random              6 位数随机验证码
* @param phoneNum            电话号码
* return result             0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestSmsAuth (String random, String phoneNum)
```

```
/**
*@Deprecated(建议不再使用此接口)已过时          发送短信验证码（手机号码作为账号进行
*注册）
* @param randno              6 位数随机验证码
* @param phoneno             电话号码
* return result             0:指令发送成功，其它数字则为指令发送失败
*/
public int smsAuth (String randno,String phoneno);
```

2.2.7 设置密码

```
/**
* 设置账号密码（完成注册/忘记密码）
* @param pwd                 密码
* @param username            用户名（手机客户端目前只针对手机号码和邮箱）
* return result             0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestSetSecret (String pwd, String username)
```

```
/**
* @Deprecated(建议不再使用此接口)已过时          设置账号密码（完成注册/忘记密码）
* @param pwd                 密码
* @param username            用户名（手机客户端目前只针对手机号码和邮箱）
* return result             0:指令发送成功，其它数字则为指令发送失败
*/
public int setSecret (String pwd,String username);
```

2.2.8 初始化用户登录、注销功能实体

```
/**
*初始化 DongAccount            建议全局只初始化一次
*@param callback               回调监听者
*/
public static void initDongAccount (DongAccountCallback callback);
```

2.2.9 判断 DongAccount 的初始化状态

```
/**
*  初始化 DongAccount          判断是否实例化 DongAccount 时调用
* @return result               true:已经初始化完,false:未初始化
*/
public static boolean initCompleteDongAccount ();
```

2.2.10 注册回调监听者

```
/**
*  注册 DongAccount     功能的回调监听者     建议用在界面 onResume 方法中
*@param impl                  被注册的回调监听者
*/
public static void registerAccountCallback (DongAccountCallbackImp impl);
```

2.2.11 解绑回调监听者

```
/**
*  解绑 DongAccount     功能的回调监听者     建议用在界面 onPause 方法中
*@param impl                  被解绑的回调监听者
*/
public static void unRegisterAccountCallback (DongAccountCallbackImp impl);
```

2.2.12 帐号登录

```
/**
*  账号登录(DongSDKProxy 类中的接口)
```

```
*@param username              登陆用户名
*@param password              登陆密码
* return result               0:指令发送成功，其它数字则为指令发送失败
*/
Public static int login (String username, String password);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时          账号登录
*@param username              登陆用户名
*@param password              登陆密码
* return result               0:指令发送成功，其它数字则为指令发送失败
*/
public int login (String username, String password);
```

2.2.13 账号注销

```
/**
*  退出账号(DongSDKProxy 类中的接口)
*/
public static void loginOut ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时      退出账号
*/
public void loginOut();
```

2.3 设备管理

注：下面未过时接口在 DongSDKProxy.java 类中， @Deprecated 标注过时接口在 DongdongAccount.java 类中。

2.3.1 请求设备列表

```
/**
*  向平台请求设备列表          建议只调用一次或少调用
* return result               0:指令发送成功，其它数字则为指令发送失败
```

```
*/
public static int requestGetDeviceListFromPlatform ()
```

```
/**
*@Deprecated(建议不再使用此接口)已过时          向平台请求设备列表
* return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public int requestDeviceList();
```

2.3.2 获取设备列表

```
/**
*  得到从平台获取的设备列表     这是从缓存中获取数据
* return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static ArrayList<DeviceInfo> requestGetDeviceListFromCache ();
```

```
/**
*@Deprecated(建议不再使用此接口)已过时          得到设备列表
* return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public ArrayList<DeviceInfo> getDeviceList ();
```

2.3.3 添加设备

```
/**
*登录用户添加设备
* @param devName              设备名称
* @param sn                   设备序列号
* return result              0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestAddDevice (String deviceName, String sn);
```

```
/**
```

```
* @Deprecated(建议不再使用此接口)已过时          给登录用户添加设备
* @param devname                设备名称
* @param sn                     设备序列号
* return result                0:指令发送成功，其它数字则为指令发送失败
*/
public int addDevice (String devname, String sn);
```

2.3.4 删除设备

```
/**
*登录用户删除设备或者取消授权账户
* @param deviceid               设备 ID
* @param userid                 设备新名称
* return result                0:指令发送成功，其它数字则为指令发送失败
*/
public int deleteDevice (int userid, int deviceid);
```

```
/**
*@Deprecated(建议不再使用此接口)已过时        登录用户删除设备或者取消授权账户
* @param deviceid               设备 ID
* @param userid                 设备新名称
* return result                0:指令发送成功，其它数字则为指令发送失败
*/
public int deleteDevice (int userid, int deviceid);
```

2.3.5 设置设备名称

```
/**
*  登录用户修改设备的名称
* @param deviceID               设备 ID
* @param deviceName             设备新名称
* return result                0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestSetDeviceName (int deviceID, String deviceName);
```

```
/**
```

```
*@Deprecated(建议不再使用此接口)已过时     登录用户修改设备的名称
* @param deviceid                设备 ID
* @param devname                 设备新名称
* return result                 0:指令发送成功，其它数字则为指令发送失败
*/
public int setDeviceName (int deviceid, String devname);
```

2.3.6 开锁

```
/**
*    开锁                       透传开锁
* @param deviceId               设备 ID
* @return result                0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestUnlock (int deviceID);
```

2.4 设备授权

注：下面未过时接口在 DongSDKProxy.java 类中， @Deprecated 标注过时接口在 DongdongAccount.java 类中。

2.4.1 获取授权列表

```
/**
*登录用户请求获取已授权的列表
* @param deviceID               设备 ID
* return result                0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestGetDeviceAuthorizeAccounts (int deviceID)
```

```
/**
*@Deprecated(建议不再使用此接口)已过时       登录用户请求获取已授权的列表
* @param deviceid               设备 ID
* return result                0:指令发送成功，其它数字则为指令发送失败
*/
public int getDeviceAuthorizeAccounts (int deviceid);
```

2.4.2 授权

```
/**
*登录用户请求将该设备授权给 user
* @param user                    用户
*@param deviceID                 设备 ID
* @注释 将该设备授权给 user
*/
public static int requestAuthorize (String user, int deviceID);
```

```
/**
*@Deprecated(建议不再使用此接口)已过时      登录用户请求将该设备授权给 user
* @param user                    用户
* @param deviceid                设备 ID
* return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public int authorize (String user, int deviceid);
```

2.4.3 删除授权

参见 2.3.4 删除设备

2.5 设备播放

注：此类新增接口在 DongSDKProxy.java 类中，@Deprecated 标注过时接口在 DongdongAccount.java 类中，在调用这些接口时必须先调用 DongSDKProxy.initDongDevice 方法。

2.5.1 初始化设备播放功能实体

```
/**
*初始化 DongDevice              建议全局只初始化一次
*@param callback                回调监听者
*/
public static void initDongDevice (DongDeviceCallback callback);
```

2.5.2 判断 DongDevice 的初始化状态

```
/**
* 初始化 DongDevice                判断是否实例化 DongDevice 时调用
* @return result                   true:已经初始化完,false:未初始化
*/
public static boolean initCompleteDongDevice ();
```

2.5.3 注册回调监听者

```
/**
* 注册 DongDevice 功能的回调监听者     建议用在界面 onResume 方法中
*@param impl                       被注册的回调监听者
*/
public static void registerDongDeviceCallback (DongDeviceCallbackImp impl);
```

2.5.4 解绑回调监听者

```
/**
* 解绑 DongDevice 功能的回调监听者     建议用在界面 onPause 方法中
*@param impl                       被解绑的回调监听者
*/
public static void unRegisterDongDeviceCallback (DongDeviceCallbackImp impl);
```

2.5.5 选择设备播放

```
/**
*登录用户播放设备
* @param context                    应用程序上下文
* @param surfaceView                播放视频用的 surfaceView 控件
* @param device                     播放的设备对象
* return result                     0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestStartPlayDevice (Context context,SurfaceView surfaceView,
                    DeviceInfo device);
```

```
/**
*@Deprecated(建议不再使用此接口)已过时          登录用户播放设备
* @param context                  从界面传入上下文
* @param surfaceView              播放视频用的 surfaceView 控件
* @param device                   播放的设备对象
* return result                   0:指令发送成功，其它数字则为指令发送失败
*/
public void startPlayDevice (Context context, SurfaceView surfaceView,DeviceInfo device);
```

2.5.6 开始播放

```
/**
*  开始播放
* @param type                     播放类型  1:音频,2:视频, 4:对讲(此处建议写 5)
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestRealtimePlay (int type);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时          开始播放
* @param type                     播放类型  1:音频.2:视频,4:对讲(此处建议写 5)
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public int realtimePlay (int type);
```

2.5.7 停止播放

```
/**
*  停止播放
* @param type                     播放类型  1:音频,2:视频,4:对讲
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestStop (int type);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时          停止播放
```

```
* @param type                播放类型  1-音频  2-视频  4-对讲
* @return result             0:指令发送成功，其它数字则为指令发送失败
*/
public int stop (int type);
```

2.5.8 停止设备播放

```
/**
*停止设备播放（停止录像、关闭手机麦克风关闭机音响）
*/
public static void requestStopDeice ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时      停止设备播放（停止录像、关闭手机麦克
*风、关闭机音响）
*/
public void stopDeice ();
```

2.5.9 打开手机麦克风

```
/**
*  打开手机麦克风
* @return result             0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestOpenPhoneMic ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时      打开手机麦克风
*/
public void openPhoneMic ();
```

2.5.10 关闭手机麦克风

```
/**
*  关闭手机麦克风
```

```
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestClosePhoneMic ();
```

```
/**
*@Deprecated(建议不再使用此接口)已过时          关闭手机麦克风
*/
public void closePhoneMic  ();
```

2.5.11 打开手机音响

```
/**
*打开手机音响
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestOpenPhoneSound ();
```

```
/**
*@Deprecated(建议不再使用此接口)已过时          打开手机音响
*/
public void openPhoneSound ();
```

2.5.12 关闭手机音响

```
/**
*  关闭手机音响
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestClosePhoneSound ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时          关闭手机音响
*/
public void closePhoneSound ();
```

2.6 设备控制

注：此类新增接口在 DongSDKProxy.java 类中，@Deprecated 标注过时接口在 DongdongAccount.java 类中，在调用这些接口时必须先调用 DongSDKProxy.initDongDeviceSetting 方法。

2.6.1 初始化设备控制功能实体

```
/**
*初始化 DongDeviceSetting    建议全局只初始化一次
*@param callback             回调监听者
*/
public static void initDongDeviceSetting (DongDeviceSettingCallback callback;
```

2.6.2 判断 DongDeviceSetting 的初始化状态

```
/**
* 初始化 DongDeviceSetting   判断是否实例化 DongDeviceSetting 时调用
* @return result             true:已经初始化完,false:未初始化
*/
public static boolean initCompleteDongDeviceSetting ();
```

2.6.3 注册回调监听者

```
/**
* 注册 DongDeviceSetting 功能的回调监听者    建议用在界面 onResume 方法中
*@param impl                 被注册的回调监听者
*/
public static void    registerDongDeviceSettingCallback (DongDeviceSettingCallbackImp
                                                         impl);
```

2.6.4 解绑回调监听者

```
/**
* 解绑 DongDeviceSetting 功能的回调监听者    建议用在界面 onPause 方法中
*@param impl                 被解绑的回调监听者
*/
```

```
public static void unRegisterDongDeviceSettingCallback (DongDeviceSettingCallbackImp
                                                                    impl);
```

2.6.5 开锁

```
/**
*    开锁                          视频监视开锁
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
Public static int requestDOControl ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        开锁
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public int lockControl ();
```

2.6.6 保存实时图片

```
/**
*  截取正在播放视频此刻的单一帧图像
* @param fileFolderName       保存 SD 卡下的根目录
* @param device               设备
*@return result               保存路径
*/
public static String requestTakePicture (String fileFolderName, DeviceInfo device)
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        得到正在播放的视频此刻单一帧图像,文
*件保存在 sd 卡的根目录 Viewer/image 里面。返回值为 false 的时候意味着截图失败，请检*
查是否有 sd 卡,或者检查您的设备是否已经打开
*/
public boolean takePicture ();
```

2.6.7 开启本地录像

```
/**
*  打开手机本地录像                暂未实现
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestOpenRecord ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        打开手机本地录像，录像的文件保存在手
*机 SD 卡根目录的 Viewer/video 文件夹下
*/
public void openRecord ();
```

2.6.8 关闭本地录像

```
/**
*  关闭手机本地录像
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestCloseRecord ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        关闭手机本地录像
*/
public void closeRecord ();
```

2.6.9 设置透传信息

```
/**
*  设置透传信息
* @param    deviceID              设备 ID
* @param data                     透传信息
*@return result                  0:指令发送成功，其它数字则为指令发送失败
*/
```

```
public static int requestSdkTunnel (int deviceID, byte[] data);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        设置透传信息
* @param    deviceid                设备 ID
* @param data                       透传信息
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public int setSdkTunnel (int deviceid, byte[] data);
```

2.6.10 获取下载外链

```
/**
*  获取设备的下载外链，详细信息见 InfoDownloadUrl 类
* @param deviceid                   设备 id
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestGetDownloadUrls (int deviceID);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时       获取设备的下载外链，详细信息见
*InfoDownloadUrl 类
* @param deviceid                   设备 id
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public int getDownloadUrls (int deviceid);
```

2.6.11 设置设备视频分辨率

```
/**
*  设置设备视频分辨率、
* @nQualityType                     0:流畅,1:标清,2:高清
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestSetQuality (int nQualityType);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        设置设备视频分辨率、
*@nQualityType                    0:流畅,1:标清,2:高清
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public int SetQuality (int nQualityType);
```

2.6.12 获取设备视频分辨率

```
/**
*  获取设备视频亮度
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestGetQuality ();
```

2.6.13 设置设备视频亮度

```
/**
*  设置设备视频亮度
*@ brightness                     范围: 1-100
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestSetBCHS (int brightness);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        设置设备视频亮度
*@ brightness                     范围: 1-100
* @return result                  0:指令发送成功，其它数字则为指令发送失败
*/
public int SetBCHS (int brightness);
```

2.6.14 获取设备视频亮度

```
/**
*  获取设备视频亮度
```

```
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestGetBCHS ();
```

2.6.15 设置设备音频大小

```
/**
*  设置设备音频大小
*@ spkVolume                       范围: 1-50  注：大于 50 按 50 处理
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestSetAudioQuality (short spkVolume)；
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        设置设备音频大小
*@ spkVolume                       范围: 1-50  注：大于 50 按 50 处理
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public int SetAudioQuality (short spkVolume)；
```

2.6.16 获取设备音频大小

```
/**
*  获取设备视频亮度
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestGetAudioQuality ();
```

2.7 局域网接口

注：此类新增接口在 DongSDKProxy.java 类中，@Deprecated 标注过时接口在 DongdongAccount.java 类中，在调用这些接口时必须先调用 DongSDKProxy.initDongDeviceSetting 方法。

2.7.1 初始化局域网功能实体

```
/**
*初始化 DongAccount            建议全局只初始化一次
*@param callback               回调监听者
*/
public static void initDongAccountLan (DongAccountCallback callback);
```

2.7.2 判断 DongAccount 的初始化状态

```
/**
*  初始化 DongAccount          判断是否实例化 DongAccount 时调用
* @return result               true:已经初始化完,false:未初始化
*/
public static boolean initCompleteDongAccountLan ();
```

2.7.3 注册回调监听者

```
/**
*  注册 DongAccount 功能的回调监听者    建议用在界面 onResume 方法中
*@param impl                   被注册的回调监听者
*/
public static void registerAccountLanCallback (DongAccountCallbackImp impl);
```

2.7.4 解绑回调监听者

```
/**
*  解绑 DongAccount 功能的回调监听者    建议用在界面 onPause 方法中
*@param impl                   被解绑的回调监听者
*/
public static void unRegisterAccountLanCallback (DongAccountCallbackImp callback);
```

2.7.5 局域网账户对象

```
/**
*局域网可以当做一个特殊的平台账户来对待。只是这个账户不需要录的，是默认的账户
```

```
*@ callback                    回调对象
*/
public static void initDongAccountLan (DongAccountCallback callback);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        局域网也可以当做一个特殊的平台账户
*来对待。只是这个账
*                              户不需要登录的，是默认的账户
*/
public DongdongAccount getDongdongAccount ();
```

2.7.6 请求设备列表

```
/**
*  请求局域网设备列表
* @return result                0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestLanDeviceListFromPlatform ();
```

```
/**
*@Deprecated(建议不再使用此接口)已过时        请求局域网设备列表
* @return result                0:指令发送成功，其它数字则为指令发送失败
*/
public int requestDeviceList ();
```

2.7.7 开始搜索局域网设备

```
/**
*  局域网搜索设备
* @return result                0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestLanStartScan ();
```

```
/**
```

```
* @Deprecated(建议不再使用此接口)已过时        局域网搜索设备
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public int LanStartScan ();
```

2.7.8 停止搜索局域网设备

注：离开局域网列表后应调用该接口停止局域网内搜索设备，因为局域网内搜索设备比较耗系统网络资源

```
/**
*  局域网停止搜索设备
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestLanStopScan ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        局域网停止搜索设备
* @return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public int LanStopScan();
```

2.7.9 刷新局域网设备列表

```
/**
*  局域网刷新设备
*@return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public static int requestLanFlush ();
```

```
/**
* @Deprecated(建议不再使用此接口)已过时        局域网刷新设备
*@return result                    0:指令发送成功，其它数字则为指令发送失败
*/
public int LanFlush();
```

2.7.10 清除局域网功能实体

```
/**
*局域网刷新设备
*/
public static void requestLanLoginOut ();
```

2.7.11 设备播放

参见 2.5 设备播放

2.7.12 设备控制

参见 2.6 设备控制

三、回调

3.1 设置回调监听

注：此类新增接口在 DongSDKProxy.java 类中， @Deprecated 标注过时接口在 DongdongAccount.java 类中。建议使用者使用 AbstractDongCallbackProxy 类中的内部类，这样可以选择性的实现自己需要的接口，详情参考 Demo 中用法。

3.1.1 设置帐户回调监听者

```
/**
*  设置 DongRegisterCallback 接口的回调监听者
* @ impl                          回调监听者
*/
public static void registerDongRegisterCallback (DongRegisterCallbackImp impl);
```

```
/**
*@Deprecated(建议不再使用此接口)已过时          设置 DongRegisterCallback 接口的回调监
*听者
*/
```

```
public void registerDongAccountCallbackListener (DongAccountCallback accountSink);
```

3.1.2 设置设备播放流程监听者

```
/**
*  设置 DongDeviceCallbackImp 接口的回调监听者
* @ impl                              回调监听者
*/
public static void registerDongDeviceCallback (DongDeviceCallbackImp impl);
```

```
/**
* @Deprecated(建议不再使用此接口)已过时          设置 DongDeviceCallbackImp 接口的回调
*监听者
*/
public void registerDongDeviceCallbackListener (DongDeviceCallback deviceSink);
```

3.1.3 设置设备设置回调监听者

```
/**
*  设置 DongDeviceSettingCallback 接口的回调监听者
* @ impl   回调监听者
*/
public static void registerDongDeviceSettingCallback (DongDeviceSettingCallbackImp impl);
```

```
/**
*@Deprecated(建议不再使用此接口)已过时     设置 DongDeviceSettingCallback 接口的回调
*监听者
*/
public void registerDongDeviceSettingCallbackListener (
                                DongDeviceSettingCallback deviceSettingSink);
```

3.2 响应回调接口

3.2.1 注册账户的回调接口

```
public interface DongRegisterCallback{

        /**
         * 调用查询该账号是否已经被注册接口[requestQueryUser (String user)]的时候回
         * 调该方法
         *@param nReason          0:未被注册过,1:已被注册过
         *@return result          自定义返回值
         */
        int OnQueryUser (int nReason);

      /**
       * 调用发送短信接口[requestSmsAuth (String random,String phoneNum)]的时候回调
       *该方法
       *@param nReason          0:成功,1:失败
       *@return result          自定义返回值
       */
        int OnSmsAuth (int nReason);

       /**
       * 调用设置密码接口[requestSetSecret (String pwd,String username)]的时候回调该方
       *法
       *@ param   nReason          0:成功,1:失败
       *@return result          自定义返回值
       */
        int OnSetSecret (int nReason);

       /** 在注册的过程中发生错误
       *@param nErrNo           发生错误的错误码,详情参见错误码定义
       *@return result          自定义返回值
       */
        int OnRegisterError (int nErrNo);
}
```

3.2.2 账户的回调接口

```
public interface DongAccountCallback {
```

```
/**
* 登录平台账户，调用登陆接口[login(String userName,String password)]的时候连
* 接服务器成功回调该方法
* @return result                自定义返回值
*/
int OnConnect ();

/**
*登录平台账户，调用登陆接口[login(String userName,String password)]的时候账
*号登陆成功回调该方法
*@param tInfo                 账号实例对象
*@return result               自定义返回值
*/
int OnAuthenticate (InfoUser tInfo);

/**
*调用请求列表接口[requestLanDeviceListFromPlatform ()]的时候，该方法会被不断
*的回调从而刷新列表
*@return result               自定义返回值
*/
int OnNewListInfo ();

/**
*当有报警过来(平台在线推送)的时候，会通过该方法回调回来报警的设备列表
*@return result               自定义返回值
*/
int OnCall (ArrayList<DeviceInfo> list);

/**
*调用添加设备接口[requestAddDevice (String devname,String sn)]的时候回调该方
*法
* @param nReason               0:成功,1:无效设备,3:已经被添加,其他-失败
* @param username              设备拥有者账号名称（仅在 nReason 为 3 时有效）
*@return result               自定义返回值
*/
int OnAddDevice (int nReason, String username);

/**
*调用删除设备或者取消授权账户接口[requestDeleteDevice (String userId,String
*deviceId)]的时候回调该方法
*@param nReason               0:成功,其他:失败
*@return result               自定义返回值
```

```
*/
int OnDelDevice (int nReason);

/**
*调用修改设备名称接口[requestSetDeviceName (int deviceid,String devname)]的时
*候回调该方法
*@param nReason              0:成功,其他:失败
*@return result              自定义返回值
*/
int OnSetDeviceName (int nReason);

/**
*获取设备授权用户列表接口[requestGetDeviceAuthorizeAccounts (int deviceid)]的
*时候回调该方法
*param    list               授权用户对象列表
*@return result              自定义返回值
*/
int OnGetDeviceUserInfo (ArrayList<InfoUser> list);

/**
*调用授权接口[requestAuthorize (String user,int deviceid)]的时候回调该方法
*param nReason               0:成功,其他:失败
*param userid                授权成功的用户 id
*@return result              自定义返回值
*/
int OnAddDeviceUser (int nReason, int userid);


/**
*同一个账号只能同时登陆一台设备，否者回调该方法
*@param time                 异常登陆时间
*@return result              自定义返回值
*/
int OnLoginOtherPlace (String time);

/**
*透传信息
*param deviceId              设备 id
*param data                  透传数据
*@return result              自定义返回值
*/
  int OnSdkTunnel (int deviceId,String data);

/**
```

```
        *获取下载外链[requestGetDownloadUrls (String user,int deviceid)]的时候回调该方
        *法
        *param deviceid              设备 Id
        *param list                  下载外链对象列表
        *@return result              自定义返回值
        */
        int OnGetDownloadUrls (int deviceid,ArrayList<InfoDownloadUrl> list);

       /**
        *透传开锁的时候回调该方
        *法
        *param result                开锁结果
        *@return result              自定义返回值
        */
        int OnOpenDoor (int result);

        /**
        *发生错误，详情参见错误码定义
        *@return result              自定义返回值
        */
        int OnUserError (int nErrNo);
    }
```

3.2.3 设备播放回调接口

```
public interface DongDeviceCallback {

        /**
        *设备连接成功调用播放设备接口[requestStartPlayDevice (Context
        *context,SurfaceView surfaceView,DeviceInfo device)]设备连接成功的时候回调该
        *方法
        *param nType                 0:未知默认值,1:音频,2:视频
        *@return result              自定义返回值
        */
        int OnConnect (int nType);

        /**
        *设备认证成功调用播放设备接口[requestStartPlayDevice (Context
        *context,SurfaceView surfaceView ,DeviceInfo device)]设备认证成功的时候回调该
        *方法
        *param nType                 0:未知默认值,1:音频,2:视频
```

```
        *@return result                自定义返回值
        */
        int OnAuthenticate (int nType);

        /**
        *视频获取成功调用播放设备接口[requestStartPlayDevice (Context
        *context,SurfaceView surfaceView ,DeviceInfo device)]获取码流成功的时候回调该
        *方法
        *@return result                自定义返回值
        */
        int OnVideoSucc ();

        /**
        *播放设备的时候上行和下行流量
        *@param upload                 上行流量
        *@param download               下行流量
        *@return result                自定义返回值
        */
        int OnTrafficStatistics (float upload, float download) ;

        /** 播放异常
        *@param nReason                异常的错误码:1:没有播放权限,2:对方已挂断,3:设备占
        * 线，无法播放,4:音频播放已被其他用户占用,5:音频当前正常播放
        *@param username               正在观看的用户名
        *@return result                自定义返回值
        */
        int OnPlayError (int nReason, String username) ;

        /**
        *发生错误
        *@param nErrNo                 发生错误的错误码,详情参见错误码定义
        *@return result                自定义返回值
        */
        int OnViewError (int nErrNo);
}
```

3.2.4 设备设置回调接口

```
public interface DongDeviceSettingCallback{

        /**
        *获得设备视频质量[]
```

```
        * param nQualityType          0:流畅,1:标清,2:高清
        *@return result               自定义返回值
        */
        int OnGetQuality (int nQualityType);

        /**
        *获得设备视频亮度
        *param nBrightness            范围:1—100
        *@return result               自定义返回值
        */
        int OnGetBCHS (int nBrightness);

        /**
        *获取设备音频大小
        * param wSpkVolume            范围:1—100
        *@return result               自定义返回值
        */
        int OnGetAudioQuality (short wSpkVolume) ;

       /**
        *透传开锁的时候回调该方
        *法
        *param result                 开锁结果
        *@return result               自定义返回值
        */
        int OnOpenDoor (int result);

        /**
        *发生错误。该接口发生的错误的时候，错误码由该函数回调
        *@param nErrNo                发生错误的错误码,详情参见错误码定义
        *@return result               自定义返回值
        */
        int OnSetupError ( int nErrNo);
}
```

四、对象

4.1 InfoUser 类

```
public class InfoUser {
        public int          userID; //  用户 ID
        public String       userName; //  用户名
        public byte[]       pwd; //  密码
        public String       urlPath; //  登陆之后打开该网址

        public InfoUser(int userID, String username, byte[] pwd,String url) {
                this. userID= userID;
                this. userName= username;
                this.  pwd = pwd;
                this. urlPath =url;
        }
}
```

4.2 DeviceInfo 类

```
public class DeviceInfo implements Serializable{
        public int          dwDeviceID; //设备的 id
        public boolean      isOnline; //设备是否在线
        public String       deviceName; //设备名
        public String       deviceSerialNO; //设备的序列号
        public int          dwCapacity; //设备的扩展信息，第 23 位：0:授权设,1:我的设
                                        //备
        public byte[]       pwd;//设备密码
        public String       apSSID;//设备的热点
        public DeviceInfo(InfoDevice infoDevice){
                this. isOnline =infoDevice. isOnline;
                this. deviceName =infoDevice. deviceName;
                this. deviceSerialNO =infoDevice. deviceSerialNO;
                this.dwCapacity=infoDevice.dwCapacity;
                this.dwDeviceID=infoDevice.dwDeviceID;
                this. pwd =infoDevice. pwd;
                this. apSSID =infoDevice. apSSID;
        }
}
```

4.3 InfoDownloadUrl 类

```
public class InfoDownloadUrl {
    public String      deviceName; // 设备名
    public String      roomValue; // 房号
    public int         nSize;  // 文件大小
    public int         nRecReason; // 存储原因
    public int         nFileType; // 文件类型 图片或视频
    public String      timeStamp; // 时间戳
    public String      url; // 下载外链

    public InfoDownloadUrl(String strDeviceName, String strRoomValue,
                           int size,int recreason, int filetype, String
                           strTimestamp, String strUrl) {
        this.deviceName = strDeviceName;
        this.roomValue = strRoomValue;
        this.nSize  = size;
        this.nRecReason = recreason;
        this.nFileType  = filetype;
        this.timeStamp = strTimestamp;
        this.url = strUrl;
    }
}
```

五、错误码定义

错误码	说明
10002	平台连接失败
10003	设备连接失败
21001	平台连接断开
20007	平台登录失败
20002/40002	无效的用户名
20003/40003	无效的密码
30002	设备连接断开

20005	服务器忙
20006	数据库服务器出错
其他	系统错误