

互动直播

Android 端集成

产品文档



腾讯云

【版权声明】

©2013-2019 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

Android 端集成

下载代码

直播接口

Android 端集成

下载代码

最近更新时间：2019-05-15 14:39:03

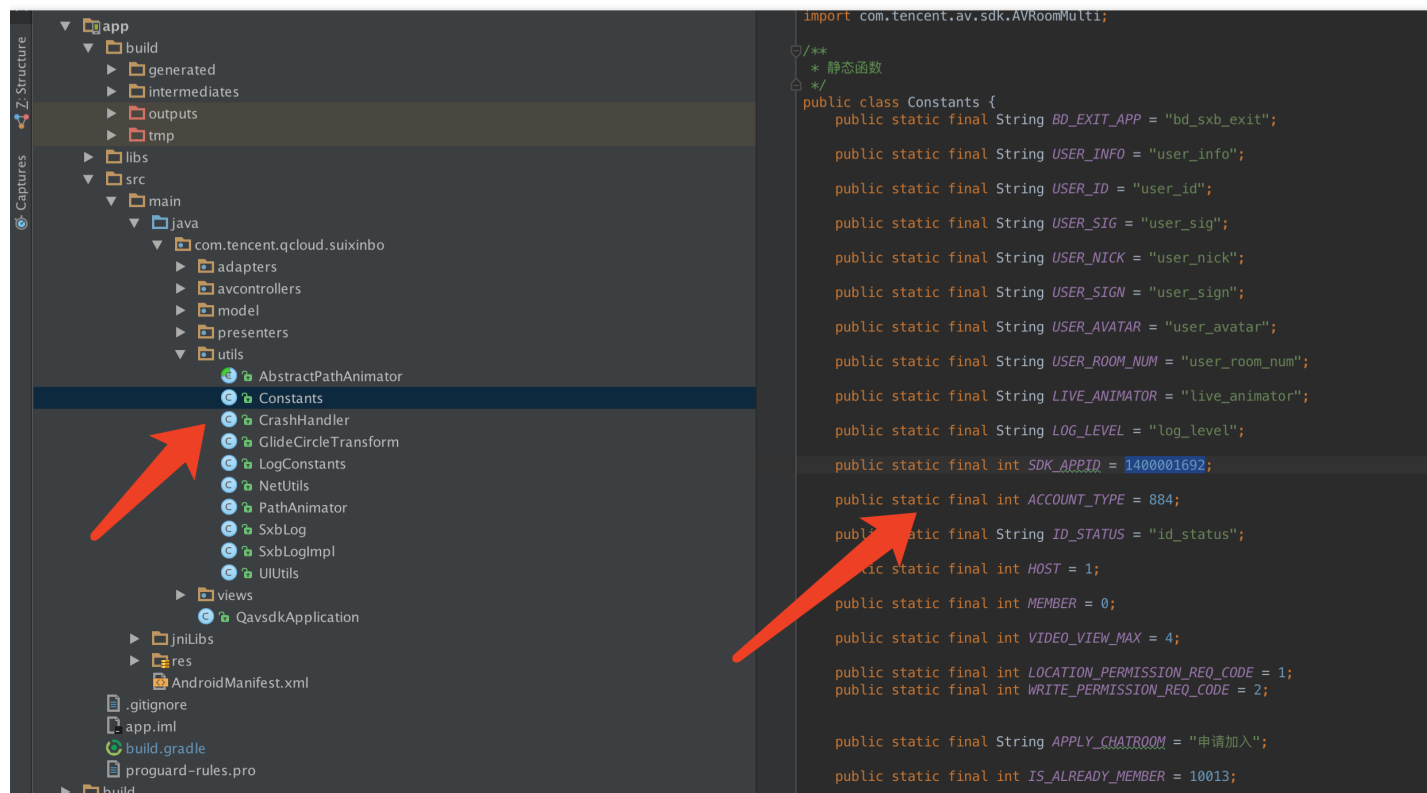
下载 Demo

目前在 GitHub 上提供了两个示例：

- [随心播](#) 演示了包括界面和后台交互的完整的直播流程。
- [简单直播](#) 最简单的互动直播示例，演示了最关键的几个接口的调用。

修改配置

将随心播代码中的 appid 和 accountType 修改成开发者自己的。



运行

编译运行工程，在启动界面选择随心播。

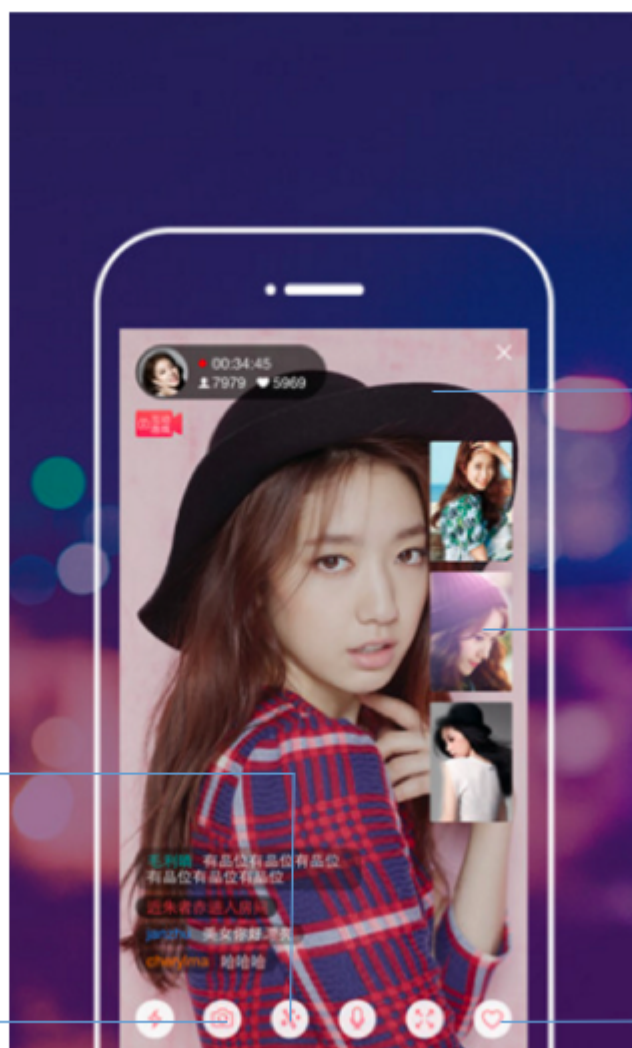


正在直播的列表

开直播按钮

美颜

切换摄像头



主播

上麦观众

点赞按钮

集成到开发者自己的代码工程里

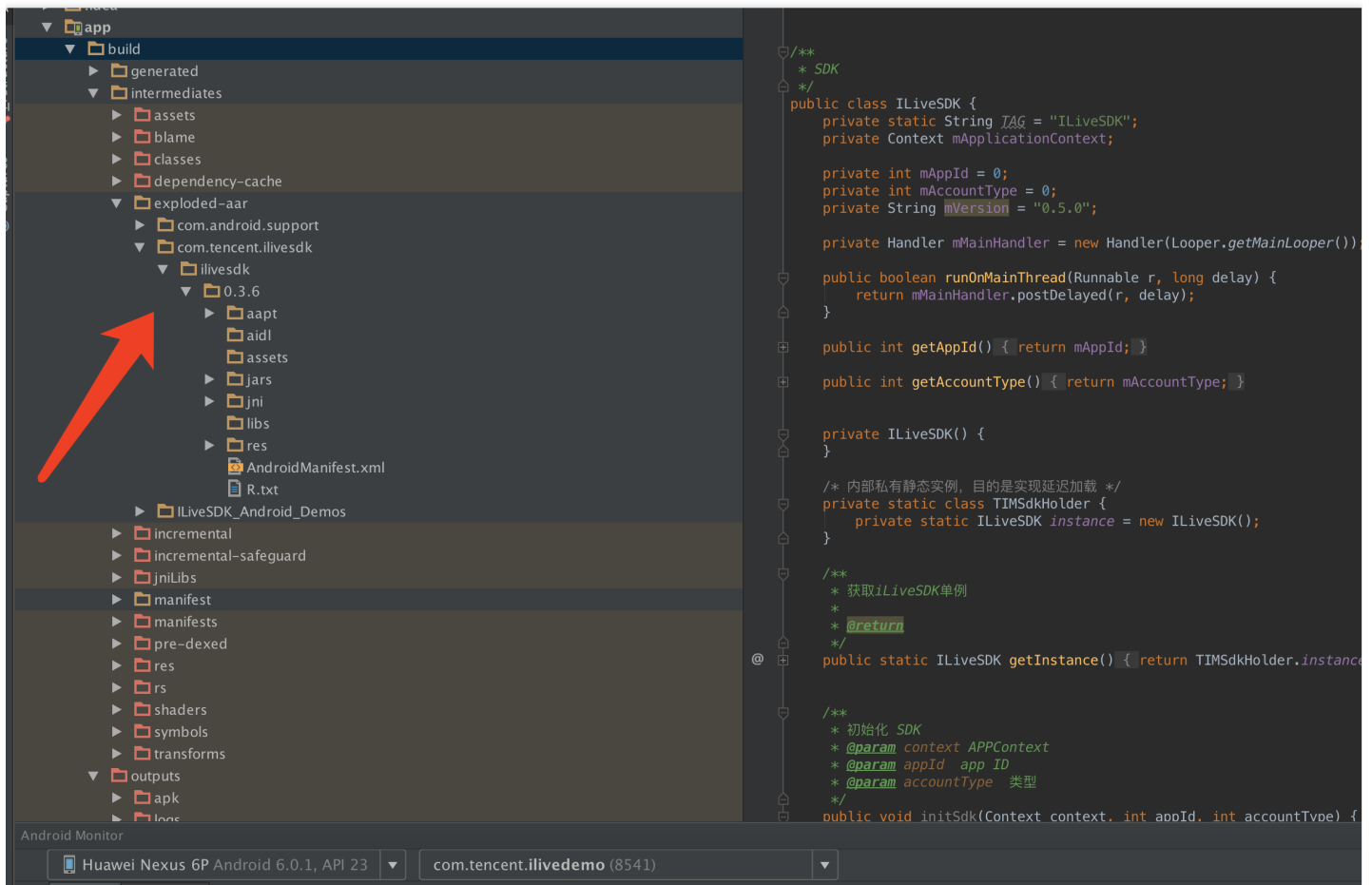
引入 SDK

- **aar 方式集成**（强烈推荐）

如果您使用的 Android Studio 开发，那么导入 iliveSDK 非常简单。只需两行代码就可以搞定了。（X.X.X 替换成对应版本号，如1.0.1）。同步完成之后可以在 build 文件夹中找到 liversdk 和 ilivesdk 文件夹。

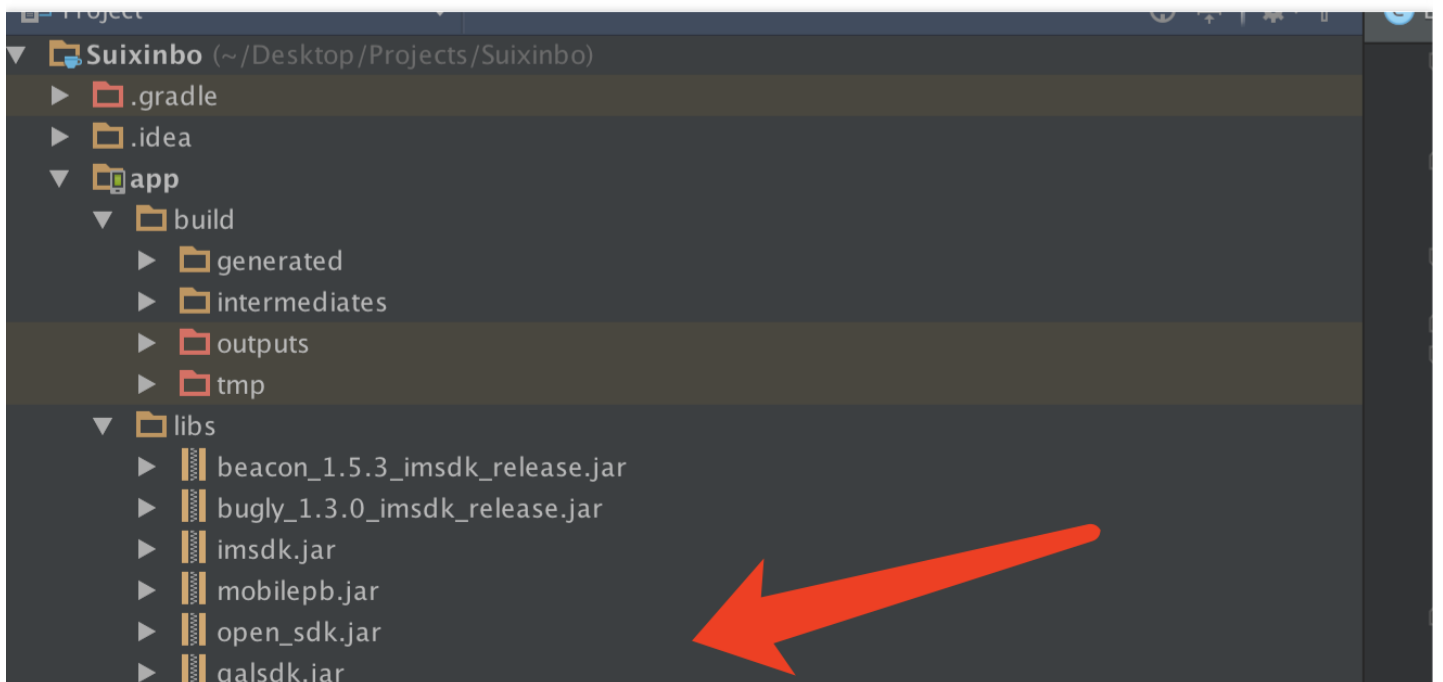
核心功能

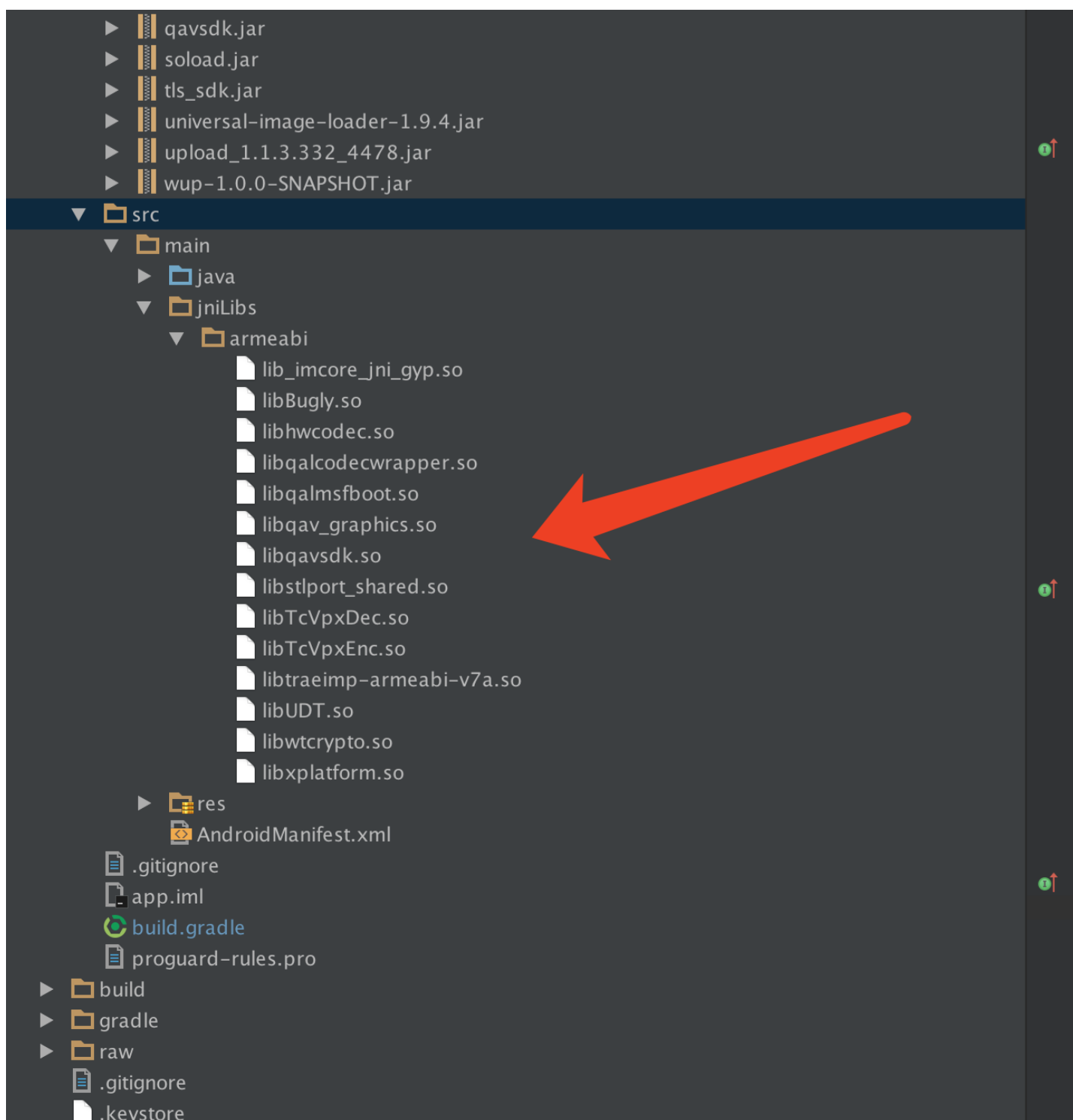
```
compile 'com.tencent.ilivesdk:ilivesdk:X.X.X'
```



• 传统库类方式集成

在腾讯云官网 [下载音视频库类](#)。需要把 so 文件和 jar 包文件分别放到对应 jnilibs 和 libs 里面。





配置服务修改后台 server 地址

目前随心播业务后台（开源可复用，[源码](#)）主要用来维护直播房间列表。如果复用随心播客户端代码，需要修改随心播后台地址为业务方自己部署的服务器地址。

接口	说明
GET_MYROOMID	获取自己分配的房间号

接口	说明
NEW_ROOM_INFO	创建新房间
STOP_ROOM	退出房间
GET_LIVELIST	获取房间列表
SEND_HEARTBEAT	房间心跳
GET_COS_SIG	图片上传相关

```

/**
 * 网络请求类
 */
public class OKhttpHelper {
    private static final String TAG = OKhttpHelper.class.getSimpleName();
    private static OKhttpHelper instance = null;
    public static final String GET_MYROOMID = "http://182.254.234.225/sxb/index.php?svc=user_av_room&cmd=get";
    public static final String NEW_ROOM_INFO = "http://182.254.234.225/sxb/index.php?svc=live&cmd=start";
    public static final String STOP_ROOM = "http://182.254.234.225/sxb/index.php?svc=live&cmd=end";
    public static final String GET_LIVELIST = "http://182.254.234.225/sxb/index.php?svc=live&cmd=list";
    public static final String SEND_HEARTBEAT = "http://182.254.234.225/sxb/index.php?svc=live&cmd=host_heartbeat";
    public static final String GET_COS_SIG = "http://182.254.234.225/sxb/index.php?svc=cos&cmd=get_sign";

    public static OKhttpHelper getInstance() {
        if (instance == null) {
            instance = new OKhttpHelper();
        }
        return instance;
    }

    public static final MediaType JSON
        = MediaType.parse("application/json; charset=utf-8");

```

添加混淆配置

```

-keep class com.tencent.**{*;}
-dontwarn com.tencent.**

-keep class Tencent.**{*;}
-dontwarn Tencent.**

-keep class qalsdk.**{*;}
-dontwarn qalsdk.**

```

配置 service

```
<!--TLS Qal 一些服务 -->
<service
    android:name="com.tencent.galsdk.service.QalService"
    android:exported="false"
    android:process=":QALSERVICE" />

<receiver
    android:name="com.tencent.galsdk.QALBroadcastReceiver"
    android:exported="false">
    <intent-filter>
        <action android:name="com.tencent.galsdk.broadcast.qal" />
    </intent-filter>
</receiver>
<receiver
    android:name="com.tencent.galsdk.core.NetConnInfoCenter"
    android:process=":QALSERVICE">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.TIME_SET" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.TIMEZONE_CHANGED" />
    </intent-filter>
</receiver>
</application>
```

配置权限

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_LOGS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />

<!--<uses-feature android:name="android.hardware.camera" />-->
```

删除非 armeabi 架构 so

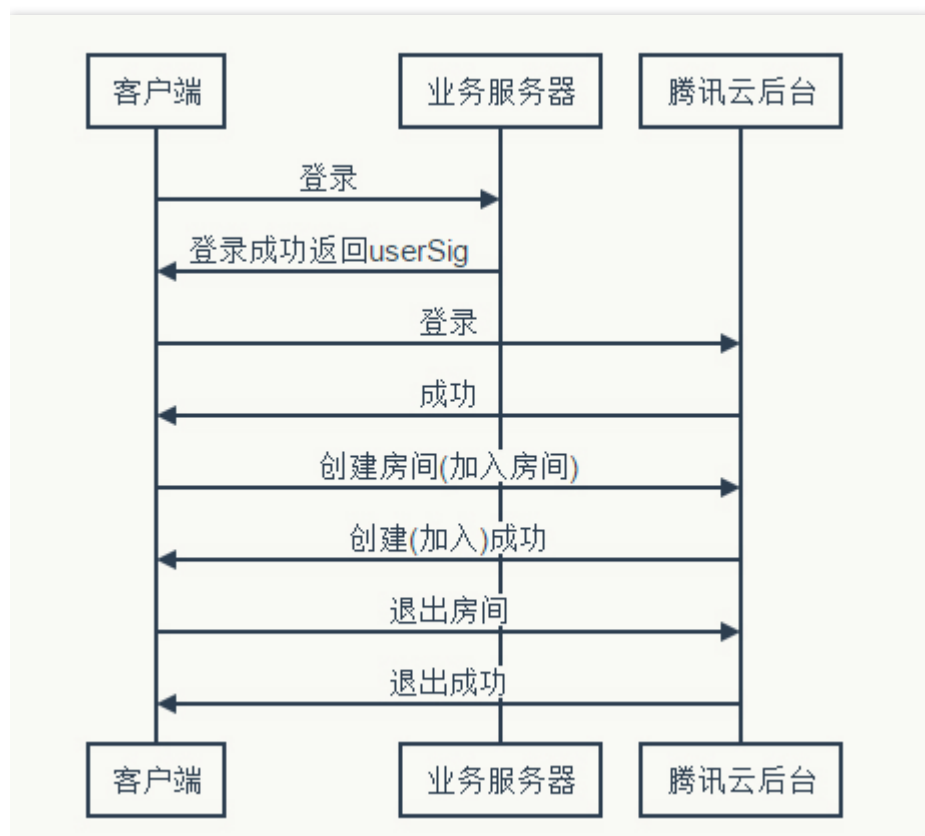
由于目前只支持 armeabi 和 armeabi-v7a 架构，如果工程(或依赖库)中有多架构，需要在 build.gradle 中添加以下配置（如果包含子工程，那么子工程中也需要添加配置）。

```
android{
    defaultConfig{
        ndk{
            abiFilters 'armeabi', 'armeabi-v7a'
        }
    }
}
```

直播接口

最近更新时间：2019-03-28 12:09:16

直播流程示例



初始化

接口名	接口描述
initSDK	iLiveSDK 的部分类的预初始化，是所有行为的第一步，告知身份传入 appId

参数类型	说明
Context	建议用 ApplicationContext
int	传入业务方 appId
int	传入业务方 accountype

```
//iLiveSDK 初始化
```

```
ILiveSDK.getInstance().initSdk(getApplicationContext(), appld, accounttype);
```

```
//初始化房间场景
```

```
ILiveRoomConfig roomConfig = new ILiveRoomConfig();
```

```
ILiveRoomManager.getInstance().init(roomConfig);
```

账号登录

接口名	接口描述
iLiveLogin	使用托管方式或独立模式，在获取到用户的 Sig 后，使用登录接口，告知后台音视频模块上线了（包括 avsdk）

参数类型	说明
String	用户 ID，在直播过程中的唯一标识
String	鉴权的密钥 Sig，如果是独立登录方式，则是业务方后台计算生成后下发的
ILiveCallBack	帐号登录回调接口，通知上线是否成功

```
ILiveLoginManager.getInstance().iLiveLogin(ILiveSDK.getInstance().getMyUserId(), "123456", new ILive  
CallBack() {  
    @Override  
    public void onSuccess(Object data) {  
        bLogin = true;  
        Toast.makeText(ContactActivity.this, "login success !", Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onError(String module, int errCode, String errMsg) {  
        Toast.makeText(ContactActivity.this, module + "login fail " + errCode + " " + errMsg, Toast.LENGTH_  
SHORT).show();  
    }  
});
```

创建房间

接口名	接口描述
createRoom	创建一个直播，只有在初始化和登录成功之后才能创建直播

参数类型	说明
int	房间 ID，房间唯一标识，建议由业务方后台统一分配
ILiveRoomOption	房间配置项，可以设置角色、权限、主播 ID 和摄像头参数等，具体参考类： ILVLiveRoomOption
ILiveCallBack	创建房间回调接口，通知创建房间是否成功

//创建房间配置项

```
ILiveRoomOption hostOption = new ILVLiveRoomOption()
```

```
.controlRole(Constants.HOST_ROLE)//角色设置
```

```
.authBits(AVRoomMulti.AUTH_BITS_DEFAULT)//权限设置
```

```
.cameralId(ILiveConstants.FRONT_CAMERA)//摄像头前置后置
```

```
.videoRecvMode(AVRoomMulti.VIDEO_RECV_MODE_SEMI_AUTO_RECV_CAMERA_VIDEO);//是否开始半  
自动接收
```

//创建房间

```
ILiveRoomManager.getInstance().createRoom(room, hostOption, new ILiveCallBack() {
```

```
@Override
```

```
public void onSuccess(Object data) {
```

```
Toast.makeText(LiveActivity.this, "create room ok", Toast.LENGTH_SHORT).show();
```

```
logoutBtn.setVisibility(View.INVISIBLE);
```

```
backBtn.setVisibility(View.VISIBLE);
```

```
}
```

```
@Override
```

```
public void onError(String module, int errCode, String errMsg) {
```

```
Toast.makeText(LiveActivity.this, module + "|create fail " + errMsg + " " + errMsg, Toast.LENGTH_SHO  
RT).show();
```

```
}
```

```
});
```

加入房间

接口名	接口描述
joinRoom	观众角色调用加入房间接口

参数类型	说明
int	房间 ID，房间唯一标识，建议由业务方后台统一分配

参数类型	说明
ILiveRoomOption	房间配置项，可以设置角色、权限、主播 ID 和摄像头参数等，具体参考类： ILiveRoomOption
ILiveCallBack	加入房间回调接口，通知加入房间是否成功

⚠ 注意：

普通观众加入房间时，应该配置 authBits 无上行权限（仅观看权限）。否则会 and 主播一样走 [核心机房DC](#)，产生高额费用。

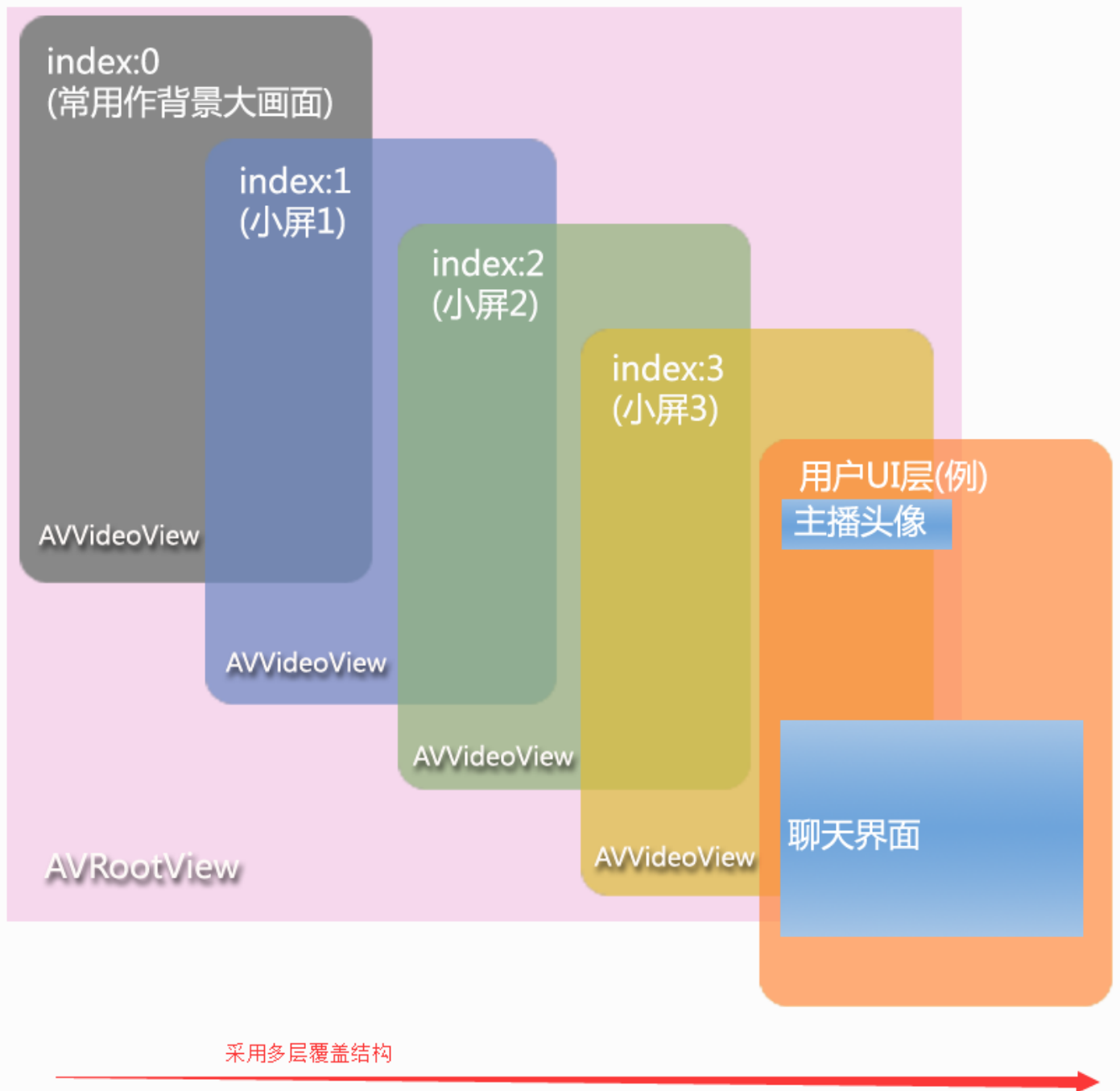
```
//加入房间配置项
ILiveRoomOption memberOption = new ILiveRoomOption("")
    .autoCamera(false) //是否自动打开摄像头
    .controlRole(Constants.NORMAL_MEMBER_ROLE) //角色设置
    .authBits(AVRoomMulti.AUTH_BITS_JOIN_ROOM | AVRoomMulti.AUTH_BITS_RECV_AUDIO | AVRoomMulti.AUTH_BITS_RECV_CAMERA_VIDEO | AVRoomMulti.AUTH_BITS_RECV_SCREEN_VIDEO) //权限设置
    .videoRecvMode(AVRoomMulti.VIDEO_RECV_MODE_SEMI_AUTO_RECV_CAMERA_VIDEO) //是否开始半自动接收
    .autoMic(false); //是否自动打开 mic
//加入房间
ILiveRoomManager.getInstance().joinRoom(room, memberOption, new ILiveCallBack() {
    @Override
    public void onSuccess(Object data) {
        bEnterRoom = true;
        Toast.makeText(LiveActivity.this, "join room ok ", Toast.LENGTH_SHORT).show();
        logoutBtn.setVisibility(View.INVISIBLE);
        backBtn.setVisibility(View.VISIBLE);
    }

    @Override
    public void onError(String module, int errCode, String errMsg) {
        Toast.makeText(LiveActivity.this, module + "join fail " + errMsg + " " + errMsg, Toast.LENGTH_SHORT).show();
    }
});
```

设置渲染层

渲染层级示例图在界面层 XML 插入一个 AVRootView，音视频数据最终通过 AVRootView 渲染出来。考虑到多屏互动情况，AVRootView 实际上不是一层 View，而是多层 AVVideoView 的叠加。直播业务默认主播在第0层（默

认最大)，其他互动观众分别在第一、第二以及第三层。每层大小都可以动态调节。



```
<com.tencent.ilivesdk.view.AVRootView
android:id="@+id/av_root_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="@color/white" />
avRootView = (AVRootView) findViewById(R.id.av_root_view);
ILiveRoomManager.getInstance().initAvRootView(avRootView);
```