

Yaokan SDK3 Android 开发文档

文件编号: YKSDK3ANDROID-20190723
版本: v1.1

深圳遥看科技有限公司
(版权所有, 切勿拷贝)

| 版本 | 说明 | 备注 | 日期 |
|------|-------|------|----------|
| v1 | 新建 | Dong | 20190723 |
| v1.1 | 增加API | Peer | 20190801 |
| | | | |

1. 概述

YaokanSDK3 提供完整的设备配网, 设备管理, 遥控器管理功能, 开箱即用, 快速与已有App对接的目的。

2. 文档阅读对象

使用 YaokanSDK3 Android 版的客户

3. 环境配置

1. 清单文件

AndroidManifest.xml 添加如下权限

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

2. 网络配置

- 在 `res` 文件夹下新建xml文件夹，创建 `network_security_config.xml` 文件，配置如下：

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

- 在 `AndroidManifest` 的 `application` 节点下添加：

```
android:networkSecurityConfig="@xml/network_security_config"
```

这是为了解决 `Android9.0` 下 `Okhttp3.0` 的bug

3. 按键对应表和数据库表配置

新建 `Assets` 文件夹，将Demo中的 `yaokan.xml` 文件拷贝到文件夹里

4. 添加遥看SDK库

在 `lib` 下添加 `yaokansdk.aar` 文件

5. gradle 配置

```
android {
    ...
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    repositories {
        flatDir{ dirs 'libs' }
    }
    configurations.all {
        resolutionStrategy.eachDependency { DependencyResolveDetails details ->
            def requested = details.requested
            if (requested.group == 'com.android.support') {
                if (!requested.name.startsWith("multidex")) {
                    details.useVersion '28.0.0' // 这里改为你项目里的版本
                }
            }
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0' // 这里改为你项目里的版本
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.0'
    implementation 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
    implementation 'com.google.code.gson:gson:2.8.2'
    implementation 'com.squareup.okhttp3:okhttp:3.11.0'
    implementation 'ch.acra:acra:4.9.0'
    implementation(name: 'yaokansdk', ext: 'aar')
}
```

4. API 列表

4.1 初始化接口

使用Appid和AppSecret进行初始化SDK，初始化成功后才能使用SDK的功能

```

class App extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        Yaokan.initialize(this);
    }
}

class YourActivity extends Activity {
    @Override
    public void onCreate() {
        super.onCreate();
        Yaokan.instance().init(getApplication(), "appId", "appSecret");
    }
}

Yaokan.instance().addSdkListener(YaokanSDKListener);

// 回调
@Override
public void onReceiveMsg(MsgType msgType, YkMessage ykMessage) {
    switch (msgType) {
        case Init:
            if (ykMessage != null && ykMessage.getCode() == 0) {
                // 初始化成功
            } else {
                // 初始化失败
            }
            break;
    }
}
}

```

4.2 设备接口

1. 配置入网

- 使用非5G Wi-Fi 网络配网

使用非5G Wi-Fi 网络配网

```

**
 * @param context 上下文
 * @param psW Wi-Fi密码
 * @param yaokanSDKListener 回调监听器
 */
Yaokan.instance().smartConfig(context,psW, yaokanSDKListener);

```

- 回调

```

@Override
public void onReceiveMsg(MsgType msgType, final YkMessage ykMessage {
    switch (msgType) {
        case StartSmartConfig:
            // 配网开始
            break;
        case SmartConfigResult:
            // 配网结果
            break;
    }
}

```

2. 获取设备列表

获取设备列表，本地发现的也会实时显示到列表中

- 获取设备方式一：字符串格式

```
String s = Yaokan.instance().exportDeviceListStringFromDB();
```

- 获取设备方式二：集合格式

```
List<YkDevice> mList = Yaokan.instance().getDeviceList();
```

3. 导入设备列表

导入设备，客户可以用来同步用户的设备列表

- 导入设备方式一：字符串格式导入

```
Yaokan.instance().inputYkDevicesToDB("[{\"did\":\"XXXXXXXXXXXXXX\",\"mac\":\"XXXXXXXXXXXX\",\"name\":\"YKK_1.0\"}]");
```

- 导入设备方式二：集合格式导入

```
if (mList.size() > 0) {
    Yaokan.instance().inputYkDevicesToDB(mList);
}
```

4. 删除设备

```
Yaokan.instance().deleteDevice(mac);
```

5. 设备测试

发送测试指令，设备收到后指示灯会快闪一次

```

/**
 *
 * @param mac 设备Mac
 * @param did 设备Did
 */
Yaokan.instance().test(mac, did);

```

6. 发送命令

◦ 常规遥控器发码

发送非空调遥控指令，包括红外遥控和射频指令

```

/**
 * 常规遥控器发码
 * @param did 设备Did
 * @param rid 遥控器Rid
 * @param key 指令名称
 * @param type 设备类型
 */
Yaokan.instance().sendCmd(did, rid, key, type);

```

◦ 空调遥控器发码

发送空调遥控指令

```

/**
 * 空调遥控器发码
 * @param did 设备Did
 * @param rid 遥控器Rid
 * @param order 遥控器指令对象
 */
Yaokan.instance().sendAirCmd(did, rid, order)

/**
 *
 * @param mode 模式
 * @param speed 风速
 * @param temp 温度
 * @param ver 上下扫风
 * @param hor 左右扫风
 */
public AirOrder(String mode, String speed, String temp, String ver, String hor)

```

◦ 有独立开关的空调遥控器的扫风发码

```

/**
 * 有独立开关的空调遥控器的扫风发码
 *
 * @param did 设备Did
 * @param rid 遥控器Rid
 * @param swing 扫风类型(上下/左右)
 * @param isOn 是否打开
 */
public void sendAirCmd(did,rid,swing,isOn)

```

```

@Override
public void onReceiveMsg(final MsgType msgType, final YkMessage ykMessage) {
    switch (msgType) {
        case SendCodeResponse:
            // 发码结果回调
            break;
    }
}

```

7. 学习红外和射频

修改已创建遥控器的按键功能

```

/**
 *
 * @param mac 设备Mac
 * @param did 设备Did
 * @param rc 遥控器对象
 * @param key 要学习的指令名称
 */
Yaokan.instance().study(mac,did,rc,key)

// 停止红外学习
Yaokan.instance().stopStudy(mac,did)

// 开始射频学习
Yaokan.instance().studyRf(mac,did,rc,key)

// 停止射频学习
Yaokan.instance().stopStudyRf(mac,did,rc)

```

```
@Override
public void onReceiveMsg(final MsgType msgType, final YkMessage ykMessage) {
    switch (msgType) {
        case StudyError:
            // 学习失败
            break;
        case StudySuccess:
            // 学习成功
            break;
    }
}
```


1. 设备开灯/关灯

控制设备的指示灯开关

```
// 开灯
Yaokan.instance().lightOn(mac,did);
// 关灯
Yaokan.instance().lightOff(mac,did);
```

2. 硬件升级OTA

OTA升级硬件固件，支持显示升级进度

```
Yaokan.instance().updateDevice(mac,did);
```

3. 获取硬件版本

获取硬件当前的版本号,显示硬件版本号，格式为 x.x.x

```
Yaokan.instance().checkDeviceVersion(mac,did);
```

```
@Override
public void onReceiveMsg(final MsgType msgType, final YkMessage ykMessage) {
    switch (msgType) {
        case otaVersion:
            // 版本信息
            break;
    }
}
```

4. 设备复位

复位设备，进入配网状态

```
Yaokan.instance().resetApple(mac,did);
```

4.2 遥控器接口

1. 获取被遥控设备类型列表

```
Yaokan.instance().getDeviceType()
```

2. 获取设备品牌

获取某遥控器类别下品牌列表

```
/**
 *
 * @param tid 设备类型
 */
Yaokan.instance().getBrandsByType(tid)
```

3. 进入一级匹配

```
/**
 *
 * @param tid 设备类型
 * @param bid 品牌ID
 */
Yaokan.instance().getMatchingResult(tid,bid)
```

4. 进入二级匹配

```
/**
 *
 * @param tid 设备类型
 * @param bid 品牌ID
 * @param gid 组Id(一级匹配接口返回,空调设备传0)
 */
Yaokan.instance().getMatchingResult(tid,bid,gid)
```

```
@Override
public void onReceiveMsg(final MsgType msgType, final YkMessage ykMessage) {
    switch (msgType) {
        case Types:
            // 设备类型列表
            break;
        case Brands:
            // 品牌列表
            break;
        case Matching:
            // 一级匹配
            break;
        case SecondMatching:
            // 二级匹配
            break;
        case RemoteInfo:
            // 遥控器详情
            break;
    }
}
```

5. 保存遥控器

匹配完成后, 可以通过遥控器id下载详情保存至本地

```
/**
 *
 * @param rc 遥控器对象
 */
Yaokan.instance().saveRc(rc);
```

6. 更新遥控器

更新遥控器信息，比如名称

```
// 更新遥控器
Yaokan.instance().updateRc(rc);
```

7. 删除遥控器

```
// 用Rid删除遥控器的方法已过期，建议使用deleteRcByUUID
Yaokan.instance().deleteRc(rid);
// UUID为本地生成的遥控器唯一ID
Yaokan.instance().deleteRcByUUID(rc.getUuid());
```

8. 获取遥控器详情

遥控器详情，用于在二级匹配中切换时调用

```
RemoteCtrl rc = Yaokan.instance().getRcData(rid);
```

9. 获取遥控器列表

```
List<RemoteCtrl> list = Yaokan.instance().getRcList();
```

10. 导出遥控器JSON数据

```
String json = Yaokan.instance().exportRcString();
```

11. 导入遥控器列表

```
Yaokan.instance().saveRcList(json);
```

4.3 SDK错误码表

主要列出了调用SDK的时候，SDK回调返回的错误码信息

| 值 | 代码 | 说明 |
|------|------------------------------|-----------|
| 0 | YKSDK_SUCC | 操作成功 |
| 1001 | YKSDK_APPID_INVALID | APPID无效 |
| 1002 | YKSDK_SMARTCONFIG_TIMEOUT | 设备配置超时 |
| 1003 | YKSDK_DEVICE_DID_INVALID | 设备 did 无效 |
| 1004 | YKSDK_DEVICE_DID_INVALID | 设备 did 无效 |
| 1005 | YKSDK_CONNECTION_TIMEOUT | 连接超时 |
| 1006 | YKSDK_CONNECTION_REFUSED | 连接被拒绝 |
| 1007 | YKSDK_CONNECTION_ERROR | 连接错误 |
| 1008 | YKSDK_CONNECTION_CLOSED | 连接被关闭 |
| 1009 | YKSDK_SSL_HANDSHAKE_FAILED | ssl 握手失败 |
| 1010 | YKSDK_INTERNET_NOT_REACHABLE | 当前外网不可达 |
| 1011 | YKSDK_NOT_INITIALIZED | SDK 未初始化 |
| 1012 | YKSDK_PERMISSION_NOT_SET | app 权限不足 |