一. 概述

容联云一键登录SDK 为移动应用提供完善的三网（移动、联通、电信）一键登录功能开发框架，屏蔽其内部复杂细节，对外提供较为简洁的 API 接口，方便第三方应用快速集成一键登录功能。

当前版本：1.1.0
下载SDK后进行解压，解压后获取：

RLOklSdk.x.x.x.arr

SDK包含一键登录和本机号码校验两个不同的功能，使用场景不一样，无需一起使用。

注意事项:

- 网络取号时候请务必开启手机流量

  1.电信只支持4G网络取号

  2.移动, 联通支持4G, 3G, 2G网络取号但在非4G网络情况下容易取号失败
- 针对双卡双待手机只取当前流量卡号

二. Demo快速体验

注:

- 确保已在容联服务端开通并申请了一键登录服务，获取到了相应的appID（即子账号Id）。
- 替换 **build.gradle(app)** 文件中 **APP_ID** 的值为申请的appID（即子账号Id），并确保此appID和应用包名相对应。

三. 接入SDK的工作配置

3.1 前期准备

- 确保您的终端设备已经开启了4G网络。
- 确保已经在容联服务端开通了一键登录服务并创建了对应的账号。

3.2 环境要求

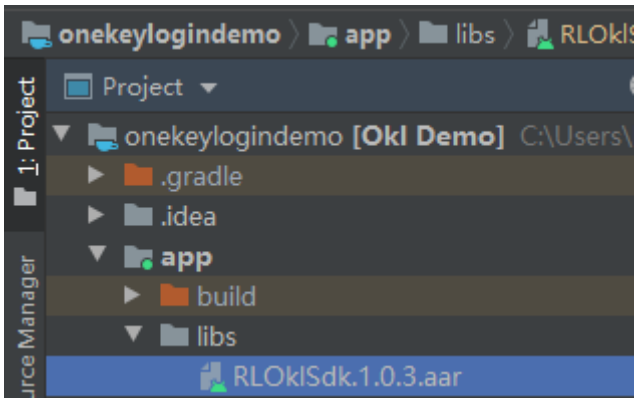
在您集成容联云 SDK 前环境要求如下:

- 建议使用Android Studio2.3及以上版本。
- JAVA 编译版本 JDK 1.7 及以上版本。
- sdk支持版本：Android4.0以上。

3.3 导入sdk（目前仅支持手动导入）

3.3.1 将解压后的RLOklSdk.x.x.x.arr文件导入您工程下的libs目录下。arr版本号以实际版本为准;

如何导入SDK，以1.0.3版本的SDK为例（其他版本SDK导入方式一致），如图所示：



3.3.2 在项目的build.gradle下添加如下代码：

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

3.3.3 手动将aar包添加依赖：

```
implementation(name: 'RLOKlSdk.x.x.x', ext: 'aar')
```

3.4 配置工程

3.4.1 权限配置

一键登录SDK需要用户允许应用程序联网，用于访问网关和认证服务器。

```
    <!-- 添加权限 -->
    <!-- 允许应用程序联网，用于访问网关和认证服务器 -->
    //允许程序访问WiFi网络状态信息
    <uses-permission android:name="android.permission.INTERNET" />
    //获取网络状态，判断是否数据、wifi等
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    //允许程序改变网络连接状态
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
    //用于判断双卡和换卡
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"
/>
    //用于判断双卡和换卡
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <!--    允许应用读写用户的外部存储器-->
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
```

注意事项:

- SDK 的权限声明与application同级，因此要放到AndroidManifest.xml 清单文件的application节点外。
- 文档仅提供了几种可选权限，开发者如需使用其他权限，可以参考Android相关文档，根据自己的应用添加所需权限。

3.4.2 SDK核心配置

清单文件配置项

```
         //以下是配置授权页   如果在只需号码验证功能可不配置
         <activity
            android:name="com.cloopen.okl.sdk.view.CmccLoginActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:launchMode="singleTop"
            android:screenOrientation="portrait" />

        <activity
            android:name="com.cloopen.okl.sdk.view.CuccLoginActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:launchMode="singleTop"
            android:screenOrientation="portrait" />

        <activity
            android:name="cn.com.chinatelecom.account.sdk.ui.AuthActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:enabled="true"
            android:exported="false"
            android:launchMode="singleTop"
            android:screenOrientation="portrait" />

        <activity

android:name="cn.com.chinatelecom.account.sdk.ui.MiniAuthActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:exported="false"
            android:launchMode="singleTop"
            android:screenOrientation="portrait" />

        <activity-alias
            android:name="com.cmic.sso.sdk.activity.LoginAuthActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:launchMode="singleTop"
            android:screenOrientation="portrait"

android:targetActivity="com.cloopen.okl.sdk.view.CmccLoginActivity" />

        <activity-alias
            android:name="com.unicom.xiaowo.account.shield.ui.LoginActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:launchMode="standard"
            android:screenOrientation="portrait"

android:targetActivity="com.cloopen.okl.sdk.view.CuccLoginActivity" />

        <activity
            android:name="com.cloopen.okl.sdk.view.OneLoginWebActivity"
```

```
            android:exported="false"
            android:launchMode="singleTop"
            android:screenOrientation="portrait" />

        <activity-alias

android:name="cn.com.chinatelecom.account.sdk.ui.PrivacyWebviewActivity"
            android:exported="false"
            android:screenOrientation="portrait"

android:targetActivity="com.cloopen.okl.sdk.view.OneLoginWebActivity" />
```

注意事项:

- SDK 的权限声明与application同级，因此要放到AndroidManifest.xml 清单文件的application节点外。

3.4.3 配置相应的签名（密钥与容联注册时提供的请保持一致）

请在build.gradle中添加相应配置：

```
//配置相应的签名  密钥与容联注册时提供的请保持一致
signingConfigs {
    release {
        storeFile file('C:\\xxx.keystore')
        storePassword '******'
        keyAlias = 'xxx'
        keyPassword '******'
    }
    debug {
        storeFile file('C:\\xxx.keystore')
        storePassword '******'
        keyAlias = 'xxx'
        keyPassword '******'
    }
}
```

3.4.4 混淆配置规则

请在工程的混淆文件中添加以下配置：

```
-dontwarn com.cloopen.**
-keep class com.cloopen.**{*;}
```

四. SDK接口说明

4.1初始化

> 使用一键登录功能前，必须使用开发者自己的 appID（即子账号Id） 进行初始化SDK
> 初始化前也可以配置SDKdebug开关打印信息

方法原型：

```
/*  init:  初始化接口
             * 参数：
             *     context - Android应用上下文对象
             *     appId     - 容联服务端申请的appId（即子账号Id）
             *     initListener - SDK初始化结果回调接口，InitListener
             *
             * 说明：示例在应用程序创建时初始化 SDK引用的是Application的上下文，
             *       开发者可根据开发需要调整。
             */
public void init(Context context, String appId, InitListener initListener)
```

示例代码：

```
        //SDK初始化（建议放在Application的onCreate方法中执行）
        OneKeyLoginHelper.getInstance().init(this, BuildConfig.APP_ID,new
InitListener() {
            @Override
            public void initStatus(String code , String data) {
                Log.e(TAG,"初始化结果,code:"+code +",data:"+data);
            }
       });
```

回调结果：(注：statusCode返回000000表示成功，具体响应码见6.4 SDK返回码)

```
code:000000,
data:
    {"statusCode":"000000",
    "statusMsg":"success",
    "msgId":"cache"
    }
```

请求参数含义：

参数名称	类型	必填	含义
context	Context	是	应用上下文环境
appId	String	是	开发者在注册应用的时候由容联服务端申请的appId（即子账号Id)
initListener	InitListener	是	InitListener为回调监听器，是一个java接口，需要调用者自己实现；InitListener是接口中的初始化回调接口，

参数名称	类型	必填	含义
			initStatus是该接口中唯一的抽象方法，即void initStatus(String code , String data)

响应结果说明：回调结果data是Json格式的字串，具体字段如下

参数名称	类型	含义
code	String	外层返回码，000000表示初始化成功，具体响应码见6.4 SDK返回码
statusCode	String	内层返回码，000000表示成功，具体响应码见6.4 SDK返回码
statusMsg	String	返回码描述说明
msgId	String	用于排查问题，如返回cache初始化成功

4.2预取号

> 预取当前运营商网络号码信息，该方法会缓存取号信息，建议在 APP 登录页初始化时调用，提高一键登录效率。调用reqPreLogin接口进行预取号，只有预取号成功才能调用sdk 的requestToken即可拿到登录的token。

方法原型：

```
/*  reqPreLogin:  预取号接口
            * 参数:
            *     timeout - 获取预取号超时时间 单位ms
            *     preLoginListener - 预取号回调监听，preLoginStatus是该监听唯一的抽象方法
            */
public void reqPreLogin(int timeout, PreLoginListener preLoginListener)
```

示例代码：

```
        //预选号
        OneKeyLoginHelper.getInstance().reqPreLogin(10000,new PreLoginListener() {
            @Override
            public void preLoginStatus(String code, String data) {
                Log.e(TAG,"preLoginStatus,code:"+code +",data:"+data);
            }
        });
```

回调结果：(注：statusCode返回000000表示成功，具体响应码见6.4 SDK返回码)

```
code:000000,
data:
    {"statusCode":"000000",
    "statusMsg":"success"
    }
```

请求参数含义：

参数名称	类型	必填	含义
timeout	int	是	获取预取号超时时间 单位ms
preLoginListener	PreLoginListener	是	PreLoginListener为回调监听器，是一个java接口，需要调用者自己实现；PreLoginListener是接口中的预取号回调接口，preLoginStatus是该接口中唯一的抽象方法，即void preLoginStatus(String code, String data)

响应结果说明：回调结果data是Json格式的字串，具体字段如下

参数名称	类型	含义
code	String	外层返回码，000000表示初始化成功，具体响应码见6.4 SDK返回码
statusCode	String	内层返回码，000000表示成功，具体响应码见6.4 SDK返回码
statusMsg	String	返回码描述说明

4.3拉起授权页

> 如果不设置setAuthUIConfig方法则展示默认的全屏
> 如果需要全屏自定义UI以及弹窗，需要在调用setAuthUIConfig授权页之前设置
> 具体设置请参照 本页 - 授权页面设计条目

方法原型：

```
/*  setAuthUIConfig:  设置自定义全屏弹窗
             * 参数:
             *     authUIConfig - 授权页配置
             */
public void setAuthUIConfig(AuthUIConfig authUIConfig)
/*  openAuth:  拉起授权
             * 参数:
             *     context - 上下文
             *     loginResultListener -授权回调
             */
public void openAuth(Context context, LoginResultListener
loginResultListener)
```

示例代码：

```
        //设置自定义全屏弹窗
        OneKeyLoginHelper.getInstance().setAuthUIConfig(authUIConfig);
         //拉起授权
         OneKeyLoginHelper.getInstance().openAuth(context,new
LoginResultListener() {
            //获取到token
            @Override
            public void loginResult(String code, String data) {
                Log.e(TAG,"login result,code:"+code + ",data:"+data);
            }
        });
```

回调结果：(注：code返回000000表示成功，具体响应码见6.4 SDK返回码)

```
code:000000,
data:
    {"resultCode":"0",
    "token":"xxxxxxxxxxxxxxxxxxxxxxxx"
    }
```

请求参数含义：

参数名称	类型	必填	含义
context	Context	是	应用上下文环境
loginResultListener	LoginResultListener	是	LoginResultListener为回调监听器，是一个java接口，需要调用者自己实现；LoginResultListener是接口中的关闭授权页回调接口，loginResult是该接口中唯一的抽象方法，即void loginResult(String code, String data)

响应结果说明：回调结果data是Json格式的字串，具体字段如下

参数名称	类型	含义
code	String	外层返回码，000000表示登录成功，具体响应码见6.4 SDK返回码
resultCode	String	内层返回码，具体响应码见6.4 SDK返回码（包含三网返回码）
resultMsg	String	返回码描述说明
token	String	登录成功获取登录临时凭证，用于获取信息接口，失败token为空

4.4退出授权页

方法原型：

```
/*  quitAuthActivity: 退出授权页
            */
public void quitAuthActivity()
```

示例代码：

```
//退出授权页
OneKeyLoginHelper.getInstance().quitAuthActivity();
```

4.5日志开关接口

方法原型：

```
/*  setDebug:  配置debug打印开关关接口
            * 参数:
            *     flag - true 打印 false不打印
            */
public void setDebug(Boolean flag)
```

示例代码：

```
// SDK配置debug打印开关
OneKeyLoginHelper.getInstance().setDebug(true);
```

请求参数含义：

参数名称	类型	必填	含义
flag	Boolean	是	配置debug是否打印，true 打印 false不打印

4.6本机号码验证

方法原型：

```
/*  phoneAuth:  本机号码验证
            * 参数:
            *     phoneAuthListener 验证回调
            */
 public void phoneAuth(PhoneAuthListener phoneAuthListener)
```

示例代码：

```
 // 本机号码验证
 OneKeyLoginHelper.getInstance().phoneAuth(new PhoneAuthListener(){
            @Override
            public void phoneAuthResult(String code, final String data) {
                Log.e(TAG,"Token of phone auth,code:"+code + ",data:"+data);
            }
        });
```

```
code:000000,
data:
    {"resultCode":"0",
    "token":"xxxxxxxxxxxxxxxxxxxxxxxx",
    "operatorType":2
    }
```

请求参数含义：

参数名称	类型	必填	含义
phoneAuthListener	PhoneAuthListener	是	PhoneAuthListener为回调监听器，是一个java接口，需要调用者自己实现；PhoneAuthListener是接口中的号码验证回调接口，phoneAuthResult是该接口中唯一的抽象方法，即void phoneAuthResult(String code, String data)

响应结果说明：回调结果data是Json格式的字串，具体字段如下

参数名称	类型	含义
code	String	外层返回码，000000表示登录成功，具体响应码见6.4 SDK返回码
resultCode	String	内层返回码，具体响应码见6.4 SDK返回码（包含三网返回码）
token	String	登录成功获取验证临时凭证，用于与服务器进行校验，失败token为空
operatorType	String	当前校验客户端类型 1-移动，2-联通，3-电信

4.6清除缓存

方法原型：

```
/*  setClearData:  配置debug打印开关关接口
            * 参数:
            *    context - Android应用上下文对象
            */
public void setClearData(Context context)
```

示例代码：

```
//清空数据
 OneKeyLoginHelper.getInstance().setClearData(Context);
```

请求参数含义：

参数名称	类型	必填	含义
context	Context	是	应用上下文环境

五. 授权页面设计

5.1授权页说明

注意：开发者不得通过任何技术手段将授权页面的隐私协议栏、slogan隐藏或者覆盖，对于接入其他认证SDK并上线的应用，我方会对上线的应用授权页面做审查，如果发现未按要求设计授权页面，将关闭应用的一键登录服务。

5.2拉起授权一键登录全屏/弹窗之前设置自定义UI

初始化默认界面UI

```
OneKeyLoginHelper.getInstance().setAuthUIConfig(AuthUIConfig configBuilder);
```

请求参数含义：

参数名称	类型	必填	含义
authUIConfig	AuthUIConfig	是	AuthUIConfig为设置授权页配置，具体设置可参考demo使用

自定义UI：调整需要自定义的参数 (可参考demo)

```
AuthUIConfig.Builder configBuilder=  new AuthUIConfig.Builder().build();
```

1.设置状态栏颜色，字体颜色 注：目前移动联通支持该配置，为了更好的页面效果，电信会跟随Manifest文件中的theme属性进行适配

setStatusBar(int statusBarColor,int navigationBarColor,boolean isLightColor)

参数	类型	说明	默认值
statusBarColor	int	自定义状态栏背景颜色	0
navigationBarColor	int	自定义底部导航栏背景颜色（没有底部导航栏的则无效）	0
isLightColor	boolean	设置状态栏内容的颜色（只能黑白），true为黑色，false为白色	false

2.设置标题栏布局 (授权页与协议页面共用)

setAuthNavLayout(int navColor, int authNavHeight)

参数	类型	说明	默认值
navColor	int	标题栏背景颜色	0xFF3973FF
authNavHeight	int	标题栏高度	49

3.设置标题栏返回按钮相关 (授权页返回相关)

setAuthNavReturnImgView(String navReturnImgPath, int returnImgWidth, int returnImgHeight, boolean navReturnImgHidden, int returnImgOffsetX)

参数	类型	说明	默认值
navReturnImgPath	String	返回按钮图片	okl_left_return
returnImgWidth	int	返回按钮图片宽度	24
returnImgHeight	int	返回按钮图片高度	24
returnImgOffsetX	int	返回按钮图片距离屏幕左边X轴偏移量	12

注：关于协议条款的标题设置返回按钮设置不同 参考4.

4.设置标题栏返回按钮相关 (协议页面返回相关)

setWebReturnImgView(String var1, int var2, int var3,int var4)

参数	类型	说明	默认值
webReturnImgPath	String	返回按钮图片	okl_left_return
webReturnImgWidth	int	返回按钮图片宽度	24
webReturnImgHeight	int	返回按钮图片高度	24
webReturnImgOffsetX	int	返回按钮图片距离屏幕左边X轴偏移量	12

5.设置标题栏中文字相关 (授权页与协议页面共用)

setAuthNavTextView(String navText, int navTextColor, int navTextSize, boolean navTextNormal, String navWebViewText, int navWebViewTextColor, int navWebViewTextSize)

参数	类型	说明	默认值
navText	String	文字设置	一键登录
navTextColor	int	字体颜色	0xFFFFFFFF
navTextSize	int	字体大小，单位为sp, 以下设置字体大小的单位与之保持 一致	17
navTextNormal	boolean	设置是否隐私条款 页面的标题栏中间文字使用默认值，true为使用navWebViewText，false为使用默认隐私条款的名字	false
navWebViewText	String	隐私条款页面的标题栏中间文字	服务条款
navWebViewTextColor	int	隐私条款页面的标题栏中间文字颜色	0xFF000000
navWebViewTextSize	int	隐私条款页面的标题栏中间文字大小	17

6.设置logo相关

setLogoImgView(String logoImgPath, int logoWidth, int logoHeight, int logoOffsetY)

参数	类型	说明	默认值
logoImgPath	String	Logo图片	ccop_one_login_logo
logoWidth	int	Logo图片宽度	71
logoHeight	int	Logo图片高度	71
logoOffsetY	int	logo相对于状态栏下边缘y偏移	125

7.设置号码相关

setNumberView(int numberColor, int numberSize, int numFieldOffsetY)

参数	类型	说明	默认值
numberColor	int	号码栏字体颜色	0xFF3D424C
numberSize	int	号码栏字体大小	24
numFieldOffsetY	int	号码栏相对于标题栏下边缘y偏移	200

8.设置Slogan相关

setSloganView(int sloganColor, int sloganSize, int sloganOffsetY, int sloganOffsetY_B, int sloganOffsetX)

参数	类型	说明	默认值
sloganColor	int	Slogan字体颜色	0xFFA8A8A8
sloganSize	int	Slogan字体大小	10
sloganOffsetY	int	Slogan相对于标题栏下边缘y偏移	382

9.设置登录按钮布局

setLogBtnLayout(String loginImgPath, int logBtnWidth, int logBtnHeight, int logBtnOffsetY, int logBtnOffsetY_B, int logBtnOffsetX)

参数	类型	说明	默认值
loginImgPath	String	登录按钮背景图片	ccop_one_login_btn_normal
logBtnWidth	int	登录按钮宽度	268
logBtnHeight	int	登录按钮高度	36
logBtnOffsetY	int	登录按钮相对于标题栏下边缘y偏移	249

10.设置登录按钮文字相关

setLogBtnTextView(String loginButtonText, int loginButtonColor, int logBtnTextSize)

参数	类型	说明	默认值
loginButtonText	String	文字设置	一键登录
loginButtonColor	int	文字颜色	0xFFFFFFFF
logBtnTextSize	int	文字大小	15

11.设置其他登录方式相关

setSwitchView(String switchText, int switchColor, int switchSize, boolean switchAccHidden, int switchAccOffsetY, OtherLoginInterface loginInterface)

参数	类型	说明	默认值
switchText	String	其他登录方式文字	切换账号
switchColor	int	其他登录方式字体颜色	0xFF3973FF
switchSize	int	其他登录方式字体大小	14
switchAccHidden	boolean	其他登录方式是否隐藏	false
		其他登录方式	

参数	类型	说明	默认值
switchAccOffsetY	int	相对于标题栏下边缘y偏移	249
loginOtherClick	OtherLoginInterface，为点击回调监听器，是一个java接口，需要调用者自己实现	设置其他登录方式点击事件	null

12.设置显示第三方登录点击事件相关(布局默认不可更改)

setIsShowOtherLogin(boolean isShowOtherLogin,OtherLoginInterface loginSmsClick, OtherLoginInterface loginWechatClick, OtherLoginInterface loginWeiboClick)

参数	类型	说明	默认值
isShowOtherLogin	boolean	设置是否显示微信、微博、QQ登录方式布局 true为显示，false不显示	false
loginSmsClick	OtherLoginInterface，为点击回调监听器，是一个java接口，需要调用者自己实现	设置qq登录点击事件	null
loginWechatClick	OtherLoginInterface	设置微信登录点击事件	null
loginWeiboClick	OtherLoginInterface	设置微博登录点击事件	null

13.设置隐私条款布局

setPrivacyLayout(int privacyLayoutWidth, int privacyOffsetY_B)

参数	类型	说明	默认值
privacyOffsetY_B	int	设置隐私条款相对于底部y偏移	18

14.设置隐私条款选择框相关

setCheckedSelectImgPath(String checkedSelectImgPath,booleanprivacyState)

参数	类型	说明	默认值
checkedSelectImgPath	string	设置选择框选择器(selector)地址	ccop_ct_auth_privacy_checkbox
privacyState	boolean	选择框是否选中	false

15.设置隐私条款字体相关

setPrivacyClauseView(int baseClauseColor, int clauseColor, int privacyClauseTextSize)

参数	类型	说明	默认值
baseClauseColor	int	设置隐私条款基础文字颜色	0xFFA8A8A8
clauseColor	int	设置隐私条款协议文字颜色	0xFF3973FF
privacyClauseTextSize	int	设置隐私条款字体大小	10

16.设置开发者隐私条款相关

setPrivacyClauseText(String clauseNameOne, String clauseUrlOne)

参数	类型	说明	默认值
clauseNameOne	String	设置隐私条款名称	null
clauseUrlOne	String	设置隐私条款URL	null

17.设置未同意隐私条款的文字提示

setPrivacyUnCheckedToastText(String privacyUnCheckedToastText)

参数	类型	说明	默认值
privacyUnCheckedToastText	String	设置未同意隐私条款的文字提示	请同意服务条款

18.设置隐私协议页面背景图片

setPrivacyBGImgPath(String privacyBGImgPath)

参数	类型	说明	默认值
privacyBGImgPath	String	设置背景图片。放在drawable 文件下	ccop_one_login_bg

19.设置授权页Activity进入动画和退出动画

setActivityTransition(int stratActivityTransition,int finishActivityTransition)

参数	类型	说明	默认值
stratActivityTransition	int	设置进入授权页动画（R.anim.xxx）	系统默认
finishActivityTransition	int	设置关闭授权页动画（R.anim.xxx）	系统默认

5.3设置弹窗显示

1.设置授权页使用对话框模式

setDialogTheme(boolean isDialogTheme, String dialogWidth, String dialogHeight)

参数	类型	说明	默认值
isDialogTheme	boolean	是否使用对话框模式	false
dialogWidth	String	授权页对话框的宽度	300
dialogHeight	String	授权页对话框的高度	500

说明：设置弹窗显示的自定义样式是通过5.2中的自定义UI进行设置（可参考demo）

六. 常见问题

1. 初始化失败

> - 检查当前手机运营商是否存在并处于有网络环境
> - 检查是否配置正确的appID（即子账号Id）
> - 检查清单文件中是否配置相应权限

2.代码编译问题及授权界面显示异常

> - 检查当前导入的arr文件是否异常
> - 检查配置清单文件中是否配置拉起activity

3.其他问题

- **包签名修改问题?**

  包名和包签名提交后不支持修改，建议直接新建应用

- **包签名不一致会有哪些影响?**

  SDK会无法使用

- **账户正常电信预选号失败**?
  检查App下build.gradle是否配置签名文件，同时删除手机项目，clean Project重新运行。

- **调试时双卡切换操作后如出现预取号异常?**

  1.检查SIM卡是否有信号，Kill掉应用，重新启动程序；

  2.wifi+4G确认wifi可连接wlan上网状态；

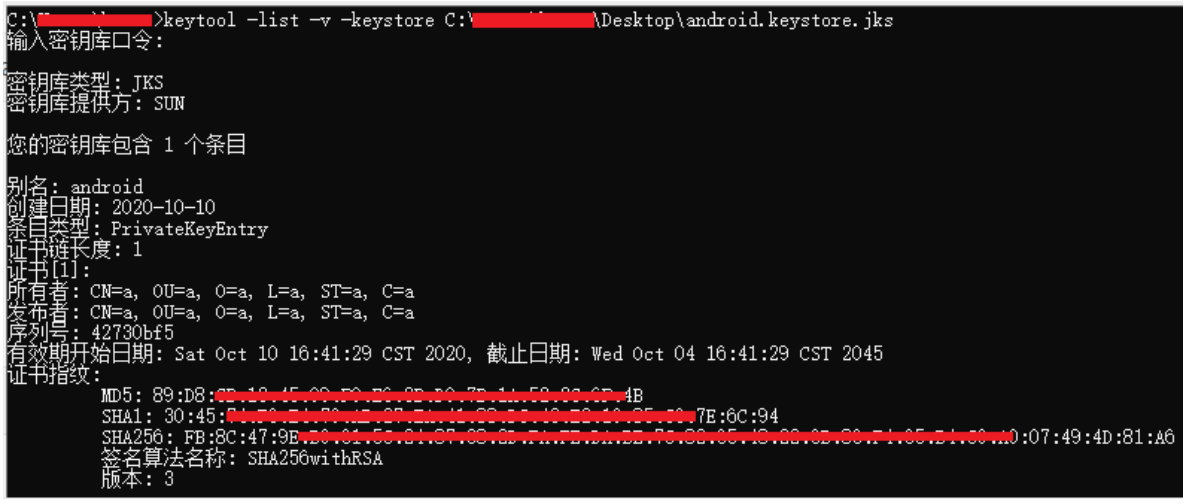
  3.检查手机是否停机。

4.Android签名获取指南

1、命令行输入以下命令：

```
keytool -list -v -keystore C:\xxx\android.keystore.jks
```

2、输入密钥库口令（密码）：



3、获取证书指纹的MD5值（去掉分隔符：":"）

样例：

```
89D83A.....8G6E5F4B
```

5.错误码

- 容联

返回码	返回码描述
000000	成功
000001	SDK初始化失败
000002	SDK未初始化
000010	授权页关闭
000011	切换账号
000013	用户取消登录
000100	应用ID为空
000101	未识别SIM卡
000102	无网络/网络异常
000103	蜂窝煤网络未开启/不稳定
000104	网络请求异常
000201	数据返回异常
000202	解析数据失败
110001	移动预取号失败
110002	移动预取号异常
110003	移动预取号响应异常
110011	移动授权失败
110021	移动验证失败
120001	联通预取号失败
120002	联通预取号异常
120003	联通预取号响应异常
120011	联通授权失败
120021	联通验证失败
130001	电信预取号失败
130002	电信预取号异常
130003	电信预取号响应异常
130011	电信授权失败
130021	电信验证失败
590000	未知错误
590001	BODY为空

返回码	返回码描述
590002	内部错误
591001	请求参数subAccId为空
591002	请求参数bundleId为空
591003	请求参数appId为空
591004	请求参数token为空
591005	请求参数os为空
591006	URI参数accountId为空
591007	accountId不合法
591008	subAccId不合法
591009	子帐号不属于主账号
591010	主账号状态不可用
591011	子账号状态不可用
591012	账户未在运营商报备
591013	运营商信息为空
591014	请求参数token不合法
591015	帐号不支持号码认证功能
591016	帐号余额不足
591019	请求参数device为空
591020	请求参数version为空
591022	请求参数bundleId不合法
591023	请求参数sign为空
591024	请求参数sign不合法
591025	请求参数random为空
591033	请求IP不在白名单内
591035	运营商应用状态不可用

- 移动

返回码	返回码描述
103000	成功
102507	登录超时（授权页点登录按钮时）
103101	请求异常
103102	包签名/Bundle ID错误（社区填写的appid和对应的包名包签名必须一致）
103111	错误的运营商请求（可能是用户正在使用代理或者运营商判断失败导致）
103119	appid不存在
103211	其他错误
103412	无效的请求（1.加密方式错误；2.非json格式；3.空请求等）
103414	参数校验异常
103511	服务器ip白名单校验失败
103811	token为空
103902	scrip失效
103911	token请求过于频繁，10分钟内获取token且未使用的数量不超过30个
104201	token已失效或不存在（重复校验或失效）
105001	联通取号失败
105002	移动取号失败
105003	电信取号失败
105012	不支持电信取号
105013	不支持联通取号
105018	token权限不足（使用了本机号码校验的token获取号码）
105021	已达当天取号限额
105302	appid不在白名单
105312	余量不足（体验版到期或套餐用完）
105313	非法请求
200005	用户未授权（READ_PHONE_STATE
200010	无法识别sim卡或没有sim卡（android）
200020	用户取消登录
200021	数据解析异常
200022	无网络
200023	请求超时

返回码	返回码描述
200024	数据网络切换失败
200025	其他错误（socket、系统未授权数据蜂窝权限等）
200027	蜂窝（数据网络）未开启或不稳定
200028	网络请求出错
200038	异网取号网络请求失败
200048	用户未安装sim卡
200050	EOF异常
200061	授权页面异常
200064	服务端返回数据异常
200072	CA根证书校验失败
200080	本机号码校验仅支持移动手机号
200082	服务器繁忙
200086	ppLocation为空
200087	仅用于监听授权页成功拉起
200089	SDK正在处理
200096	当前网络不支持取号

- 联通

返回码	返回码描述
0	处理成功
100	应用未授权
101	应用秘钥错误
102	应用无效
103	应用未授权该 IP 访问
104	应用访问次数不足
105	应用包名不正确
106	应用状态非法
107	商户状态非法
108	商户请求次数超限额
200	tokenId 无效
201	token 已失效
202	token 未授权该应用访问
203	登录鉴权级别不满足接口鉴权要求
300	接口未开放
301	应用未授权码访问该接口
302	IP 未授权码访问该接口
303	应用访问接口次数超日限额
400	请求参数为空
401	请求参数不完整
402	请求参数非法
600	请求非法
1000	请求解析错误
1001	请求已失效
1002	验签失败
1003	授权码已过期
1004	加密方式不支持
1005	RSA 加密错误
1010	服务间访问失败
1011	服务间访问错误

返回码	返回码描述
2004	用户不存在
3001	unikey 无效
3002	跳转异网取号
3003	本网执行取号失败,不需要重定向
3004	NET 取号失败
3005	上网方式为 WIFI，无法取号
3006	urlencode 编码失败
3007	请求认证接口异常
3008	msi 取号失败
3009	非联通号码
3010	网关取号错误
3011	源 IP 鉴权失败
3012	网关取号失败
3013	电信网关取号失败
3014	电信网关取号错误
3015	获取 accessCode 请求参数失败
3016	移动网关取号失败
3017	移动网关取号错误
3050	取号网关内部错误
3057	网关鉴权码查找号码失败
3058	网关鉴权码格式错误
3059	网关鉴权码已失效
3060	网关账号认证失败
3061	网关取号配额不足
3062	IP 未授权访问网关
3063	网关并发连接数受限
3064	访问网关参数非法
3065	未授权访问该网关能力
3066	网关服务暂时不可用

- 电信

返回码	返回码描述
0	请求成功
-64	权限不足
65	API-request-rates-Exceed-Limitations(调用接口超限)
-10001	调用失败
-10002	参数错误
-10003	解密失败
10004	ip 受限
10005	异网取号回调参数异常
-10006	授权失败，且属于电信网络
-10007	重定向到异网取号
-10008	超过预设取号阀值
-10009	时间戳过期
-20005	sign-invalid(签名错误)
-20006	应用不存在
-20007	公钥数据不存在
-20100	内部解析错误
-20102	加密参数解析失败
-30001	时间戳非法
-30003	topClass 失效
51002	参数为空
51114	无法获取手机号数据
80000	请求超时
80001	请求网络异常
80002	响应码错误
80003	无网络连接
80004	移动网络未开启
80005	Socket 超时异常
80006	域名解析异常
80007	IO 异常
80008	No route to host

返回码	返回码描述
80009	nodename nor servname provided, or not known
80010	Socket closed by remote peer
80100	登录结果为空
80101	登录结果异常
80102	预登录异常
80103	SDK 未初始化
80104	未调用预登录接口
80105	加载 nib 文件异常
80200	用户关闭界面
80201	其他登录方式
80800	WIFI 切换异常
80801	WIFI 切换超时