

云通讯平台 Android 开发指南

- 1 概述
 - 1.1 介绍
 - 1.2 开发流程
 - 1.3 文档术语
 - 1.4 参考文档
- 2 VoIP快速体验
 - 2.1 申请测试帐号
 - 2.2 环境搭建
 - 2.3 Demo程序介绍
 - 2.4 导入Demo工程
 - 2.5 配置帐号信息
- 3 创建自己的VoIP应用
 - 3.1 SDK介绍
 - 3.2 创建工程
 - 3.2.1 新建工程
 - 3.2.2 导入CCP SDK
 - 3.2.3 配置工程信息
 - 3.3 编写代码
 - 3.3.1 CCP SDK 初始化
 - 3.3.2 注册VoIP帐号
 - 3.3.3 设置主叫信息
 - 3.3.4 创建VoIP免费通话(或电话直拨)连接
 - 3.3.5 创建VoIP回拨呼叫连接
 - 3.3.6 接收Voip通话呼入
 - 3.3.7 接听Voip通话
 - 3.3.8 被叫挂断(或被叫拒接)VoIP通话
 - 3.4 编译运行和测试
 - 3.4.1 真机调试
 - 3.4.2 真机运行
 - 3.5 查看日志
 - 3.6 发布安装包
 - 3.6.1 工程混淆
 - 3.6.2 打包
 - 3.6.3 签名

1 概述

云通讯平台旨在为第三方应用开发者提供丰富完善的注册流程、接入机制、安全策略、管理后台以及不同语言的SDK开发包，为开发者在应用内快速、高效、低成本集成语音业务提供了一站式的服务。本文档旨在为第三方应用开发者在Android平台下集成CCP Android SDK来打造语音业务提供参考，文档预期的读者为第三方应用开发人员、平台开发人员、相关技术人员等。

1.1 介绍

云通讯平台SDK提供了网络通话、视频通话、实时对讲、聊天室、视频会议等基础能力，REST API除了提供上述功能外，还提供注册账号、创建子账号、营销外呼、语音验证码、各类查询等等。

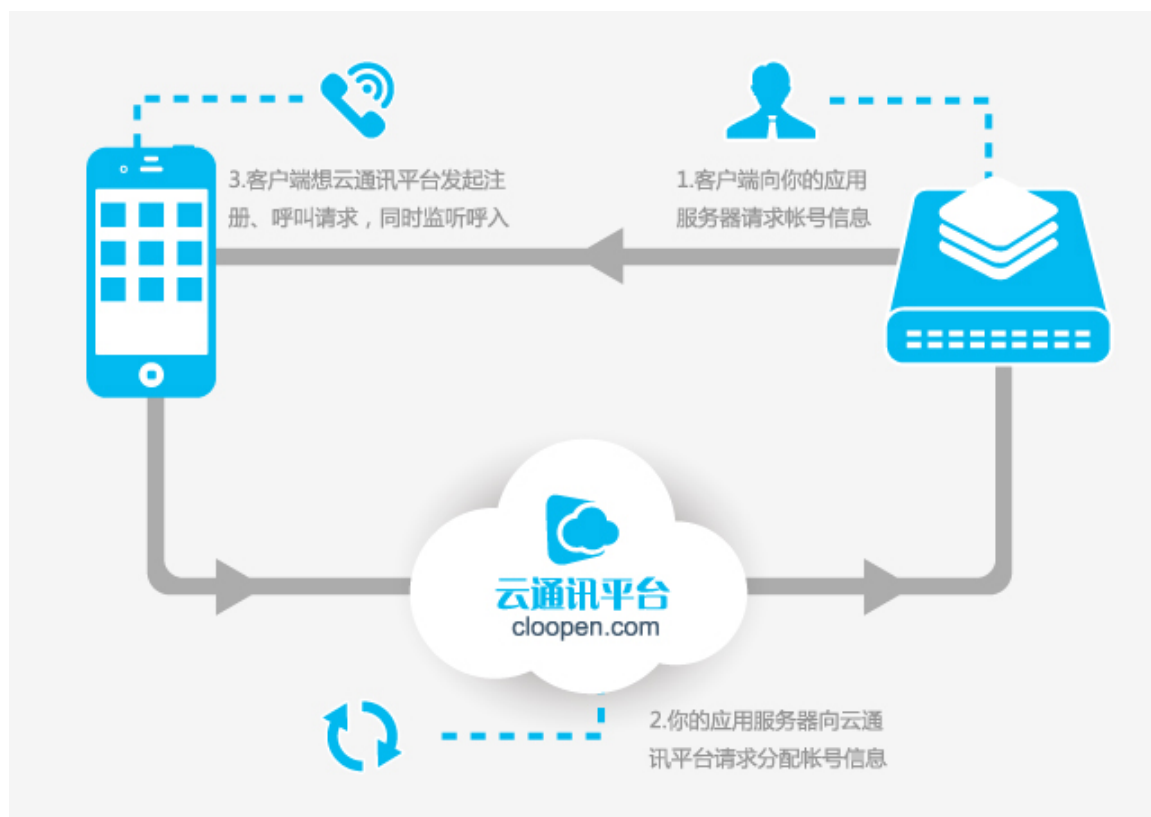
- 云通讯平台Android SDK 以Java libs的方式提供给Android平台开发人员。
- 云通讯平台iOS SDK 以C++静态库的方式提供给iOS平台开发人员。
- REST API 可通过HTTPS GET、POST方式访问。

本文档的目的就是介绍CCP SDK ANDROID平台提供的能力，并很快集成到第三方开发的应用中，轻松实现VoIP通话、发送消息的功能。

1.2 开发流程

云通讯平台作为通讯能力的云计算PAAS平台，将传统电信网络的通讯能力、基于IP的通讯能力，通过开放API以及SDK的方式提供给开发者和商家，协助开发者快速、高效、低成本打造融合通讯能力的产品。

云通讯平台能力开发方式，如下图所示。



这是常见的接入方式，通过3个主要步骤来完成：

1. 您的客户端应用集成云通讯平台提供的SDK，同时客户端向您的应用服务器请求分配VoIP账号信息；
2. 您的应用服务器通过调用云通讯平台REST API 得到用户账号并返回给您的客户端应用；
3. 客户端应用通过调用SDK API发起呼叫请求或者监听呼入；

1.3 文档术语

- AS: Application Server, 应用服务器，第三方开发者搭建的服务器，和云通讯平台交互，可以查询管理账户，也可以拨打电话、回拨电话、发送短信等。

- CCP: Cloud Communication Platform, 云通讯平台。
- CCP SDK: CCP Software Development Kit, 云通讯平台软件开发包。
- MD5: Message Digest Algorithm MD5, 消息摘要算法第五版, 为计算机安全领域广泛使用的一种散列函数, 用以提供消息的完整性保护。
- QML: Quick Markup Language, 快速标记语言, 一组当接收到来电或短信时告诉云通讯平台如何处理的指令。
- Rest: REpresentational State Transfer, 表征状态转移, 是一种针对网络应用的设计和开发方式, 可以降低开发的复杂性, 提高系统的可伸缩性。
- Rest服务器: 为应用服务器提供功能接口的服务器。
- VoIP: Voice over Internet Protocol, 基于网络协议的语音实时传输。
- VoIP帐号: 由VoIP服务器为子帐号分配的帐号。
- 开发者: 特指云通讯平台应用的第三方开发者。
- 主帐号: 第三方开发者在云通讯平台开发者网站上注册后分配得到的账号。
- 子帐号: 第三方开发者可使用主帐号调用REST接口获取的账号。

1.4 参考文档

- 《云通讯平台REST技术文档》
- 《云通讯平台Android技术文档》
- 《云通讯平台IOS技术文档》

2 VoIP快速体验

在云通讯平台注册账号, 创建Demo账号, 并下载获取CCPVoiPDemo程序(具体过程请参考以下内容)。在Demo程序中, 演示了云通讯平台提供的基础VoIP通话功能。打开工程后, 可以快速体验云通讯平台提供的VoIP通话功能。

2.1 申请测试帐号

在云通讯平台进行注册, 注册之后创建Demo, 即可获得开发VoIP所需的测试帐号信息。

测试帐号信息内容有: 主帐号、主帐号密码、子帐号、子帐号密码、VoIP帐号、VoIP帐号密码, 应用ID等。

2.2 环境搭建

- JDK1.6及以上版本, [下载](#)。
- 下载安装Eclipse, [下载](#)。
- 下载安装Android SDK, [下载](#)。
- Eclipse ADT (Android Developer Tools) 插件, SDK中包含开发Android应用所需的开发、调试工具和开发的API包., [下载](#)。
- 特别注意CCP SDK要求Android SDK最小版本是API Level 10 (即Android 2.3.3)。

2.3 Demo程序介绍

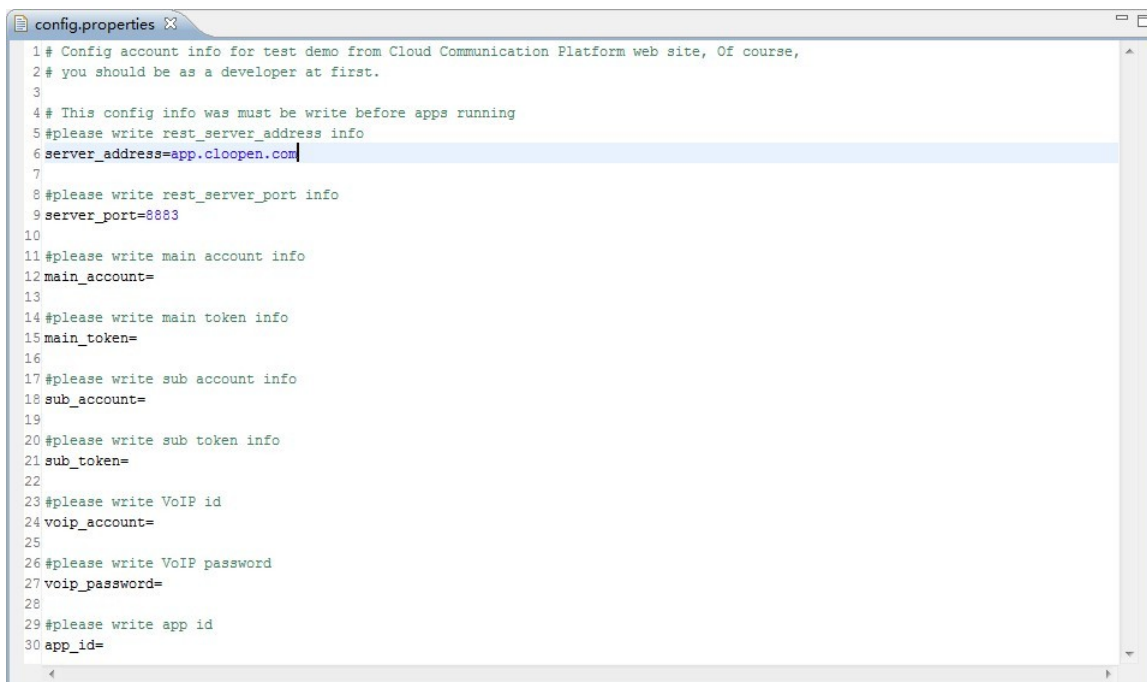
1. 在Demo账号信息页面, 提供了Android和IOS平台下的Demo下载, 请选择[Android版下载](#)
2. CCPVoipDemo功能介绍, Demo演示了CCP SDK的API接口调用, 主要实现的功能:
 - 免费电话: 需要对方的VoIP账号, 双方进行的网络P2P通话, 免费通话
 - 电话直拨: 需要对方的手机号, 主叫接入网络电话, 被叫接入普通电话的网络通话
 - 回拨呼叫: 需要对方的手机号, 双方都会接入普通电话网络进行通话
3. CCPVoipDemo程序结构说明: Demo是一个完整的Android工程, 其中主要有res、libs、src三个文件夹和AndroidManifest.xml文件构成, 下面讲解各文件的作用
 - res 工程资源文件目录, 包含图片、文字等资源。
 - libs 工程依赖第三方类库文件夹, 其中CCP核心类库就在其中, 里面还包含一个文件夹armeabi, 存放了libserphone.so文件。
 - src 工程源码
 - AndroidManifest.xml 工程配置清单文件。

2.4 导入Demo工程

1. 将下载的CCP_PHONE_DEMO_Android.rar程序解压到本地任一目录。
2. 打开Eclipse, 选择File > Import > General -> Existing Projects Into Workspace -> 点击Browse选择刚才解压所得的CCPVoipDemo工程, 点击finish成功导入项目。

2.5 配置帐号信息

1. 打开res\raw\config.properties文件, 将2.1 申请测试帐号时获取的Demo账号信息, 依次输入配置文件中, 如图所示:



```
1 # Config account info for test demo from Cloud Communication Platform web site, Of course,
2 # you should be as a developer at first.
3
4 # This config info was must be write before apps running
5 #please write rest_server_address info
6 server_address=app.cloopen.com
7
8 #please write rest_server_port info
9 server_port=8883
10
11 #please write main account info
12 main_account=
13
14 #please write main token info
15 main_token=
16
17 #please write sub account info
18 sub_account=
19
20 #please write sub token info
21 sub_token=
22
23 #please write VoIP id
24 voip_account=
25
26 #please write VoIP password
27 voip_password=
28
29 #please write app id
30 app_id=
```

注意事项:

- 为了测试VoIP的通话功能, 在获取的Demo账号信息中, 提供了两个子帐号信息, 分别填写相对应的位置, 然后多个账号之间以英文的逗号","隔开
- 环境地址配置: 沙盒环境, 请填写sandboxapp.cloopen.com; 正式环境, 填写app.cloopen.com。一定不要加前缀https或http等。

3 创建自己的VoIP应用

创建一个具有VoIP功能的应用程序,需要如下几个步骤:

1. 将CCP_SDK_v*.jar 和libserphone.so拷贝到自己创建的工程中的libs目录下。(最新版的SDK, 可参考3.1SDK介绍从网站进行下载。)
2. 在程序中对CCP SDK 进行初始化, 并利用从云通讯平台获得的帐号进行注册。
3. 在程序提供VoIP通话功能和直拨以及回拨功能的入口, 定制自己的个性化通话界面, 对通话的不同状态进行处理。

开发者可以参考我们为您提供的CCPVoipDemo程序, 帮助您快速的完成一个具有VoIP功能的应用程序。下面介绍如何集成CCP SDK, 创建自己的具有VoIP网络通话功能的应用。

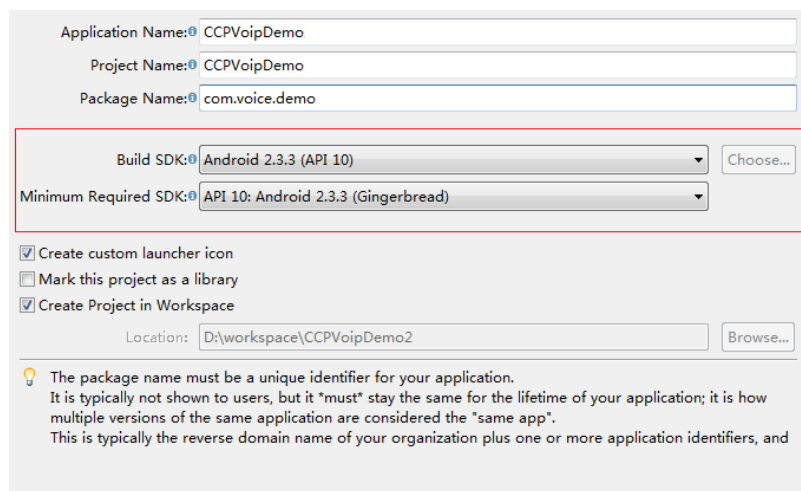
3.1 SDK介绍

1. SDK下载: 从云通讯网站下载VoIP的Android SDK, (未注册用户请先进行注册方可下载)
2. SDK文件说明: 解压下载后的压缩包到本地目录, 里面包含集成VoIP功能的CCP_SDK_v*.jar以及libserphone.so文件。这两个文件是开发者开发时候需要使用到, 放在创建的工程文件夹下:
 - CCP_SDK_v*.jar(每个第三方应用都必须导入该SDK, 实现与云通讯平台的通信)
 - libserphone.so 支持硬件驱动
3. SDK的使用方法请参考3.1.2导入CCP SDK。

3.2 创建工程

3.2.1 新建工程

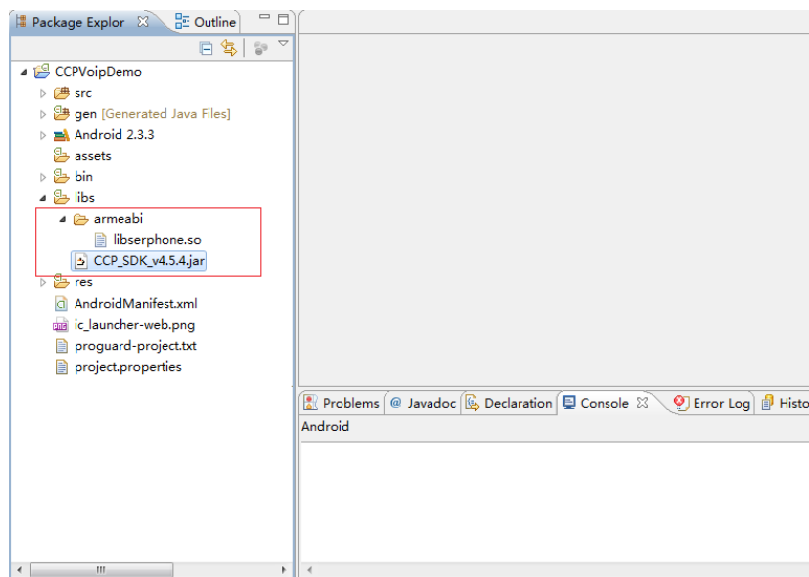
在Eclipse中建立你的工程



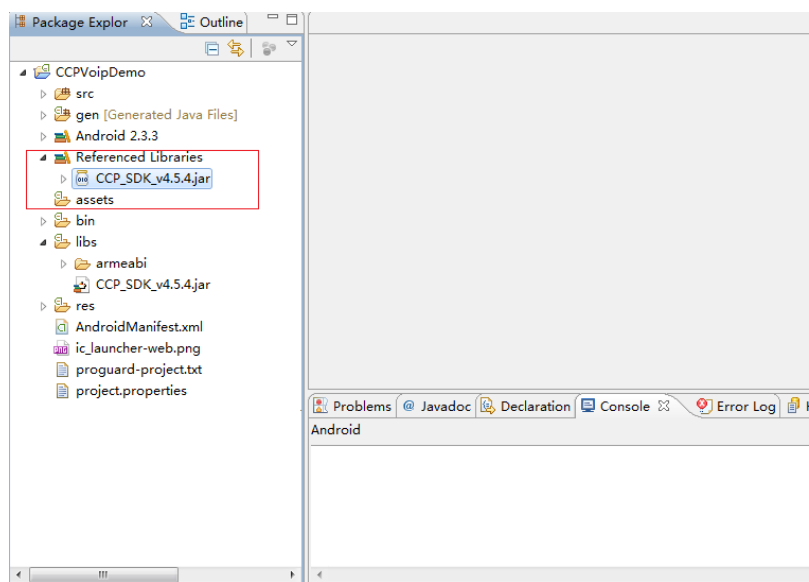
注意:在选择最低SDK版本的时候需要选择Android 2.3.3以及以上版本。(如下图所示)。其他可以不做要求然后点击下一步完成工程的创建。

3.2.2 导入CCP SDK

在工程中新建一个libs目录, 将开发工具包中libs目录下的CCP_Android_SDK_v2.2r.jar复制到该目录中。并在该级目录下新建一个armeabi目录, 将开发工具包中的Libserphone.so复制到该目录中。(如下图所示, 建立了一个名为CCPVoiDemo的工程, 并把jar包复制到libs目录下)



右键单击工程，选择Build Path中的Configure Build Path..., 选中Libraries这个tab，并通过Add Jars...导入工程libs目录下的libammsdk.jar文件。(如下图所示)。



3.2.3 配置工程信息

- 在清单文件中对上面所述呼入呼出界面进行配置，需要将jar包中的一个com.hisun.phone.core.voice.CCPService也应该进行配置
- 配置最低使用SDK信息

```
<uses-sdk android:minSdkVersion="10" android:targetSdkVersion="15" />
```

2、配置程序所需权限信息

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
<uses-permission android:name="android.permission.VIBRATE" />
```

3、配置程序服务

```
<service android:name="com.hisun.phone.core.voice.CCPService" android:exported="false" />
```

3.3 编写代码

3.3.1 CCP SDK 初始化

1. 要使你的程序启动后能支持VoIP功能，必须在代码中对CCP SDK进行初始化并且向云通讯平台注册你的帐号信息。（如下代码所示，可以在程序入口Activity的onCreate回调函数处，或其他合适的地方将你的帐号信息册到云通讯平台。注册函数示例如下代码所示。

```
// 需要传入上下文对象, 注册结果的回调
// 当CCP初始化结束后, CCP
SDK会将初始化结果通过回调结果返回给开发者，开发者可根据回调结果进行处理
CCPCall.init(getApplicationContext(), this/* InitListener */);
```

2. 如果用发起初始化函数所在的Activity接受SDK注册结果,需要使该Activity实现CCPCall.InitListener接口，这样SDK初始化结束后，就会将初始化结果返回给开发者。（如下代码所示）。

```
public class ShowAccountActivity extends Activity implements CCPCall.InitListener{

    @Override
    public void onError(Exception arg0){
        // CCP SDK注册失败回调此方法,
    }

    @Override
    public void onInitialized(){
        // CCP SDK 初始化成功回调此方法, 开发者可在该处向云通讯平台注册你的帐号信息
    }
}
```

3.3.2 注册VoIP帐号

1. 在成功初始化CCP

SDK, SDK会回调CCPCall.InitListener接口的onInitialized方法, 开发者可以在此方法内部通过帐号信息向云通讯平台发起注册请求。(如下代码所示)。

```
@Override
public void onInitialized(){
    ...
    // CCP SDK 初始化成功回调此方法, 开发者可在该处向云通讯平台注册你的帐号信息
    // 封装参数
    Map<String, String> params = new HashMap<String, String>();
    /* REST服务器地址
    params.put(UserAgentConfig.KEY_IP, "");
    /* REST服务器端口
    params.put(UserAgentConfig.KEY_PORT, "");
    /* VOIP账号, 可以填入CCP网站Demo管理中的测试VOIP账号信息
    params.put(UserAgentConfig.KEY_SID, "");
    /* VOIP账号密码, 可以填入CCP网站Demo管理中的测试VOIP账号密码
    params.put(UserAgentConfig.KEY_PWD, "");
    /* 子账号, 可以填入CCP网站Demo管理中的测试子账号信息
    params.put(UserAgentConfig.KEY_SUBID, "");
    /* 子账号密码, 可以填入CCP网站Demo管理中的测试子账号密码
    params.put(UserAgentConfig.KEY_SUBPWD, "");
    // User-Agent
    params.put(UserAgentConfig.KEY_UA, "");

    // 创建Device
    // 参数信息,能力令牌以及注册结果的回调监听
    device = CCPCall.createDevice(this/* DeviceListener *//, params);
    ....
}
```

上面代码中, 利用开发者帐号信息通过CCPCall.createDevice方法与云通讯平台的服务器完成认证, 建立连接, 并同时创建一个与服务器端通信的Device实例对象, 并且设置当收到一个CCP VoIP呼入请求时, 启动Android组件的PendingIntent。

2. 如若需要知道云平台的注册结果，需要实现DeviceListener接口。（如下代码所示，我们用发起注册请求的Activity接受请求结果。）

```
// 实现CCP SDK 初始化结果回调监听
// 实现注册结果回调监听
public class CCPHelper extends Activity implements CCPCall.InitListener ,DeviceListener{
```

3. 当CCP SDK注册结束后，SDK会回调如下两个方法通知开发者注册结果。（如下代码所示）

```
@Override
public void onConnected(){
    // 成功注册上云通讯服务器
    // TODO
}
@Override
public void onDisconnect(Reason reason){
    // 注册失败,与云通讯服务器失去连接
    // TODO
}
```

完成上面步骤，并且SDK成功注册上了云通讯服务器即可获得Device对象(Device可用于发起一次VoIP网络通话，返回该次通话的CallId,用此ID可对该次通话进行挂断等操作)。

3.3.3 设置主叫信息

- 当SDK完成初始化并且利用帐号信息成功注册上云通讯服务器，即可通过Device实例对象，在VoIP呼出通话时，传递简单的参数给被叫，如主叫手机号码和昵称。

```
// 设置主叫手机号(被叫界面显示)
device.setSelfPhoneNumber("");
// 设置主叫昵称(被叫界面显示)
device.setSelfUserName("");
```

现在，完成了上述操作，您已经可以正常使用SDK为您提供的各种功能了，

3.3.4 创建VoIP免费通话(或电话直拨)连接

通过Device对象可以发起一次VoIP通话，如下代码中通过getDevice获取与服务器建立连接的Device实例对象，调用makeCall，创建一个呼出连接，通话成功建立将返回当前通话的callid。

1. 发起一次VoIP通话

```
// 发起一次VoIP免费通话
// 参数为通话类型和对方的VoIP帐号
mCurrentCallId = getDevice().makeCall(Device.CallType.VOICE, "");
```

mCurrentCallId：主要功能是可以主动挂断本次通话使用，VoIP
通话是通过callid区分不同VoIP通话线路，Device.CallType.VOICE:语音通话，第二个参数:对方的VoIP帐号

注意事项:

- 如果在创建一个呼出连接时, 获取Device为null, 说明开发者在注册到我们服务器参数有问题, 注册失败了, 请重新检查参数, 重新连接我们服务器。
- 呼叫对方Voip帐号失败, 请检查对方Voip帐号是否输入错误或对方是否已经注册到我们的服务器。
- 如果返回的mCurrentCallId为null, 则可能SDK初始化失败。

现在您已经成功创建了一个呼出连接, 并且通过Device.makeCall, 获取到本次通话的mCurrentCallId, 您可以调用Device.releaseCall方法, 挂断本次通话。(如下代码所示, getDevice()为封装的获取Device对象)

```
// 挂断本次VoIP通话,传入本次通话的消息ID
// 第二个为预留参数,默认为0
getDevice().releaseCall(mCurrentCallId, 0);
```

2. 发起一次直拨通话

```
// 号码
private String mPhoneNumber;

// 发起一次直拨通话
// 参数为通话类型和对方的电话号码
mCurrentCallId = getDevice().makeCall(Device.CallType.VOICE,
    VoiceUtil.getStandardMDN(mPhoneNumber));
```

mCurrentCallId:主要功能是可以主动挂断本次通话使用, VoIP
通话是通过callid区分不同VoIP通话线路, Device.CallType.VOICE:直拨通话, mPhoneNumber:对方的电话号码

3. Voip通话回调方法介绍

SDK收到一个VoIP通话时, SDK会回调DeviceListener接口中的方法, 我们只需要在实现了DeviceListener接口的类中处理本次通话即可, (VoIP 通话是通过callid区分不同VoIP通话线路)。

- 与服务器建立连接, 回调, 参数为当前通话mCurrentCallId标示:

```
@Override
public void onCallProceeding(String callId){
    // 与云通讯服务器建立连接, 返回本次通话连接的通话id
    // TODO
}
```

- 呼叫到对方(对方接收到VoIP通话请求), 回调, 参数为当前通话callid标示:

```
@Override
public void onCallAlerting(String callId){
    // 对方收到VoIP,可以处理响铃等操作
    // TODO
}
```

- 对方接受VoIP通话回调, 参数为当前通话callid标示:

```
@Override
public void onCallAnswered(String callId){
    // 对方接受VoIP通话
    // TODO
}
```

- 呼叫对方失败(网络不好或对方不在线)，回调参数为当前通话callid标示，失败原因：

```
@Override
public void onMakeCallFailed(String callId, Reason reason){
    // VoIP呼叫失败,返回本次通话的ID和失败的原因
    // TODO
}
```

- 呼叫对方接受后挂断本次通话，回调参数为当前通话callid标示

```
@Override
public void onCallReleased(String callId){
    // 对方挂断本次VoIP通话
    // TODO
}
```

3.3.5 创建VoIP回拨呼叫连接

1. 发起一次回拨通话

```
// 号码
private String mPhoneNumber;

// 发起一次直拨通话
// 传入本机号码和对方的电话号码
getDevice().makeCallBack(CCPCConfig.Src_phone, mPhoneNumber);
```

CCPCConfig.Src_phone:本机号码，mPhoneNumber:对方的电话号码
注意事项：

- 如果在创建一个呼出连接时，获取Device为null，说明开发者在注册到我们服务器参数有问题，注册失败了，请重新检查参数，重新连接我们服务器，
- 呼叫对方VoIP帐号失败，请检查对方VoIP帐号是否输入错误或对方是否已经注册到我们的服务器。

3.3.6 接收Voip通话呼入

在SDK注册成功并且创建Device对象时设置了呼入参数，作为呼入事件。（如下代码所示）

1. 设置呼入参数，收到VoIP通话时唤起通话界面。

```
// 设置当呼入请求到达时, 唤起的界面
// 此处设置当接收到呼入连接时, 启动CallInActivity(替换自己创建的呼入Activity) 界面
Intent intent = new Intent(this, CallInActivity.class);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
device.setIncomingIntent(pendingIntent);
```

2. 当被叫手机收到VoIP通话请求时候,SDK会唤起通话界面, 并且CCPSDK将当前通话的callId以Intent的Bundle参数传递给在该界面。（如下代码所示）。

```
// 号码
private String mPhoneNumber;
// 通话 ID
private String mCurrentCallId;
...
Intent intent = getIntent();
Bundle extras = intent.getExtras();
...
mVoipAccount = extras.getString(Device.CALLER);
mCurrentCallId = extras.getString(Device.CALLID);
// 判断传入数据是否有误
...
```

通过如上代码可以获取到本次通话的通话ID, 以及来电者的昵称和来电号码。

3. 被叫获取Voip参数传递(可选)

- 当被叫手机收到VoIP通话请求时候,SDK会唤起通话界面, 如果主叫设置了传递参数, 被叫可以通过Intent获取主叫传递的主叫手机号和昵称。（如下代码所示）。

```
// 名称
private String mNickName;
// voip 账号
private String mVoipAccount;
...
//设置呼叫者的姓名和手机号码
String[] infos = extras.getStringArray(Device.REMOTE);
if (infos != null && infos.length > 0){
    for (String str : infos){
        if (str.startsWith(KEY_TEL)){
            // 呼叫者的电话号码
            mPhoneNumber = VoiceUtil.getLastwords(str, "=");
        } else if (str.startsWith(KEY_NAME)){
            // 呼叫者的昵称
            mNickName = VoiceUtil.getLastwords(str, "=");
        }
    }
}
```

3.3.7 接听Voip通话

1. 当被叫手机收到VoIP通话请求时候,SDK会唤起通话界面, 可以获取本次通话连接的callId, 接受本次连接, 调用Device.acceptCall(callId)方法建立通话。(如下代码所示, getDevice()为封装的获取Device对象)。

```
private OnClickListener listener = new OnClickListener(){
    @Override
    public void onClick(View v){
        try{
            // 需要判断当前的通话ID是否为空
            ...
            getDevice().acceptCall(mCurrentCallId);
            // TODO
        } catch (Exception e){
            e.printStackTrace();
        }
    }
};
```

3.3.8 被叫挂断(或被叫拒接)VoIP通话

1. 当被叫手机收到VoIP通话请求时候,SDK会唤起通话界面, 可以获取本次通话连接的callId, 可以获取本次通话连接的callId, 拒绝本次VoIP通话连接, 调用Device.rejectCall(callId)方法建立通话。(如下代码所示, getDevice()为封装的获取Device对象)。

```
private OnClickListener listener = new OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            // 需要判断当前的通话ID是否为空
            ....
            // 拒接本次VoIP通话,传入本次通话的消息ID
            // 第二个为预留参数,默认为0
            getDevice().rejectCall(mCurrentCallId,0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
};
```

2. 被叫挂断已经接受的Voip通话

- 当被叫手机收到VoIP通话请求时候,SDK会唤起通话界面, 可以获得本次通话连接的callId, 可以获得本次通话连接的callId, 接受通话之后, 挂断本次VoIP通话连接, 调用Device.releaseCall(callId, 0)方法挂断本次通话。(如下代码所示, getDevice()为封装的获取Device对象)。

```
private OnClickListener listener = new OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            // 需要判断当前的通话ID是否为空
            ...
            // 拒接本次VoIP通话,传入本次通话的消息ID
            // 第二个为预留参数,默认为0
            getDevice().releaseCall (mCurrentCallId, 0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
};
```

至此, 你已经能使用CCP Android SDK包的API内容了。如果想更详细了解每个API函数的用法, 请自行下载阅读CCPVoiDemo源码。

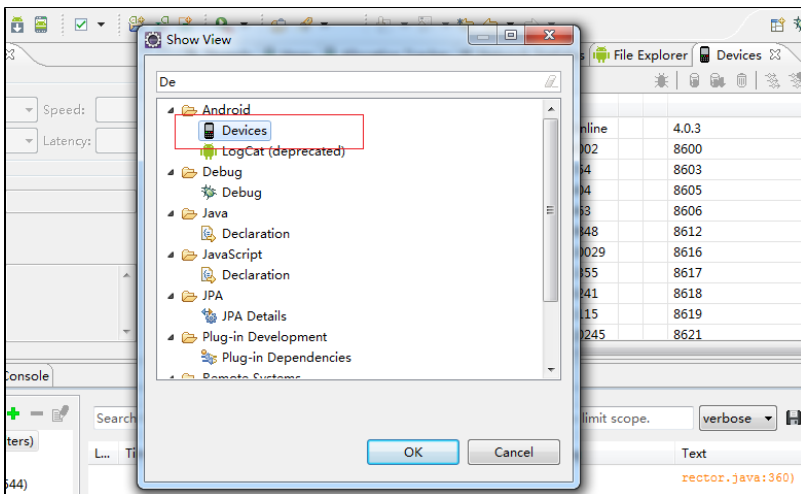
3.4 编译运行和测试

3.4.1 真机调试

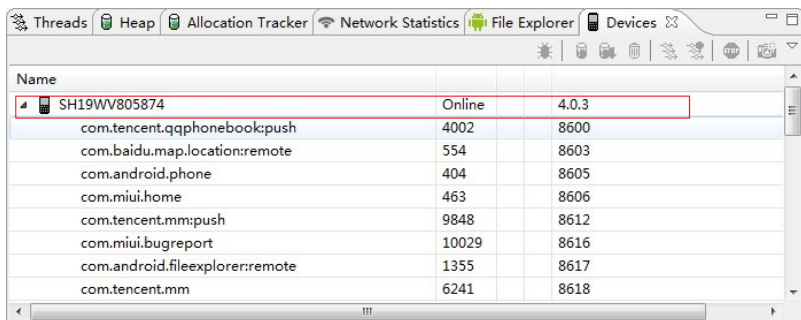
- 设置android手机为USB调试模式。步骤: menu -> 设置 -> 应用程序 -> 开发, 选择[USB调试]
- 用USB连接手机和电脑, 并确保成功。步骤: 在windows命令窗口下执行adb devices, 查看手机是否已经连接成功。
- 设置应用程序为调试模式。编辑AndroidManifest.xml 增加调试参数android:debuggable="true"

3.4.2 真机运行

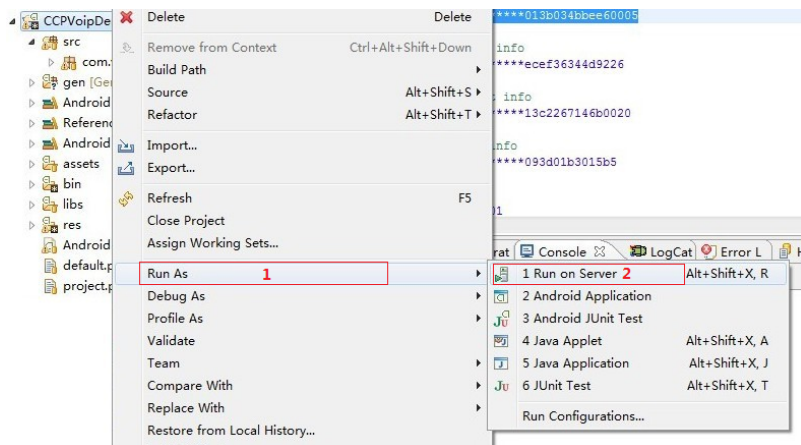
1. 在ADT插件的Devices界面选中真机。步骤Window—>Show View—>Other...—>Android—>Devices



2. 在打开的Device视图选中连接的手机设备



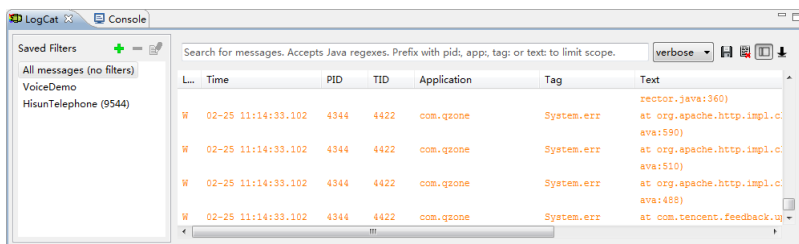
3. 右键工程点击工程—>右键—>Run As—>Android Application, 即可运行在手机上



3.5 查看日志

1. Eclipse可视化工具查看

步骤Window—>Show View—>Other...—>Android—>Logcat,即可在开发工具中查看手机输出的日志，如下所示:



2. 过滤日志信息，在Eclipse

ADT插件的Logcat界面添加一个过滤日志的Tag，如填写VoiceDemo，选择VoiceDemo过滤项查看程序demo日志信息。(VoiceDemo标签必须为程序中日志输出的TAG)。

3.6 发布安装包

3.6.1 工程混淆

当您的程序需要进行混淆的时候，为了保证SDK的正常使用，请勿将CCP SDK进行混淆，以免影响正常使用。

3.6.2 打包

编译Android工程后，在bin目录下将自动生成APK文件(Android Package)，其后缀名为".apk"。

3.6.3 签名

Android系统要求具有开发者签名的私人密钥的应用程序才可以安装。通过Eclipse中的签名向导可以方便快捷的完成生成数字签名。(查看官方文档 <http://developer.android.com/tools/publishing/app-signing.html>)。