**云之讯IM SDK**

**Android版使用说明书**

深圳市云之讯网络技术有限公司

http://ucpaas.com/front/images1/logo.png

**版权与免责声明**

**版权所有 深圳市云之讯网络技术有限公司 2015。 保留一切权利。**

非经本公司许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

本文档内容不定期更新，请用户随时关注官网的更新信息，进行正确操作。云之讯不对因为使用不当，或者人为损坏此文件的直接、 间接、特殊、附带的行为或者由之产生的损失（包括但不限于经济损失，个人损失，财产及精神损失）承担责任。

**深圳市云之讯网络技术有限公司**

地址： 深圳市南山区科技园高新南四道8号创维半导体设计大厦东座19楼

网址： <http://www.ucpaas.com/>

技术支持： 400-097-0020

合作邮箱： service@ucpaas.com

目录

[1 Android SDK导入 1](#_Toc427178785)

[1.1 集成前准备 1](#_Toc427178786)

[1.2 下载SDK 1](#_Toc427178787)

[1.3 导入SDK 1](#_Toc427178788)

[1.4 添加SDK权限和IM核心服务 2](#_Toc427178789)

[2 连接平台和注销 4](#_Toc427178790)

[2.1 初始化IM服务 4](#_Toc427178791)

[2.2 连接平台 4](#_Toc427178792)

[2.3 注销 5](#_Toc427178793)

[3 基本功能集成 6](#_Toc427178794)

[3.1 获取IMManager类实例 6](#_Toc427178795)

[3.2 会话操作 6](#_Toc427178796)

[3.3 消息操作 8](#_Toc427178797)

[3.4 讨论组操作 10](#_Toc427178798)

[3.5 群组操作 13](#_Toc427178799)

[4 对象类属性介绍 13](#_Toc427178800)

[4.1 会话类 13](#_Toc427178801)

[4.2 消息类 13](#_Toc427178802)

[4.3 讨论组信息类 14](#_Toc427178803)

[4.4 群组信息类 14](#_Toc427178804)

[4.5 返回值类 14](#_Toc427178805)

[4.6 会话类型枚举类 14](#_Toc427178806)

[4.7 消息类型枚举类 14](#_Toc427178807)

# Android SDK导入

## 集成前准备

注册并创建应用。

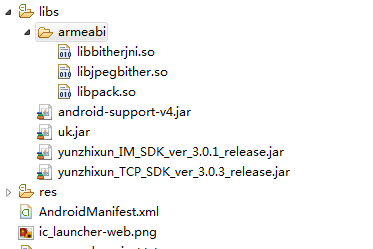
## 下载SDK

前往云之讯官网下载云之讯IMSDK，SDK包括2个jar包和1个lib库

1.IM核心功能 jar包：yunzhixun\_IM\_SDK\_ver\_x.x.x\_release.jar

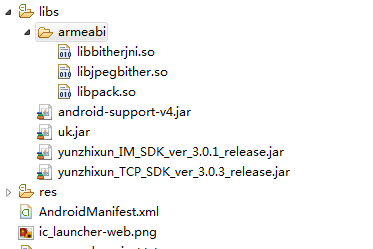
2.TCP连接jar包：yunzhixun\_TCP\_SDK\_ver\_x.x.x\_release.jar

3.传输协议处理库：libpack.so

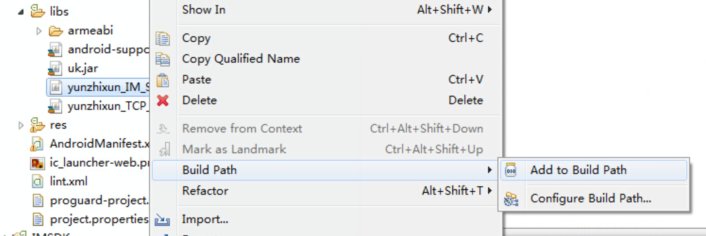


## 导入SDK

1. 将SDK包中提供的yunzhixun\_IM\_SDK\_ver\_x.x.x\_release.jar、 yunzhixun\_TCP\_SDK\_ver\_x.x.x\_release.jar包拷贝到android工程的libs目录下，将libpack.so拷贝到android工程的libs目录下的armeabi目录下，放置位置如下图所示：

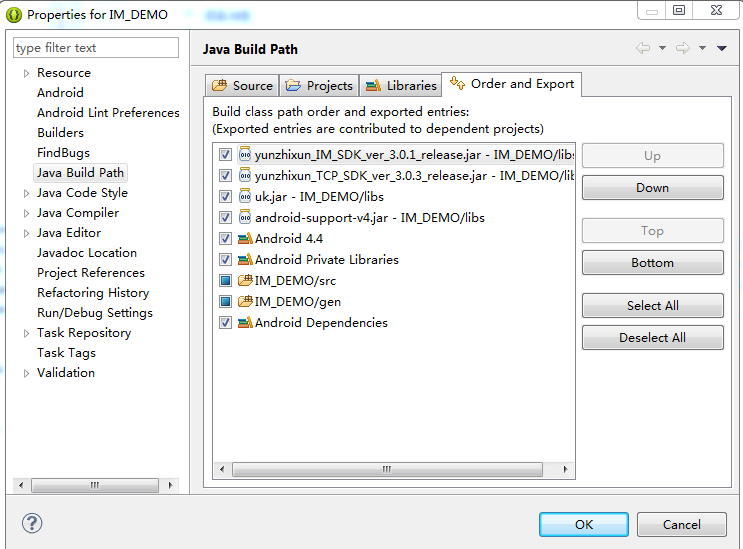


2． jar加入到Build Path中,如下图：



3． 把导入包勾选

如果执行了Add to Build Path 仍然无法使用jar包，请指向自己的工程，点击鼠标右键，选择Properties目录—>选择Java BuildPath目录—>order and Export下把导入的包勾选—>按下



## 添加SDK权限和IM核心服务

将下列权限配置添加到Android工程中的AndroidManifest文件中

*<manifest xmlns:android="http://schemas.android.com/apk/res/android"*

*package="com.yzx.im\_demo"*

*android:versionCode="1"*

*android:versionName="1.0" >*

*<uses-sdk*

*android:minSdkVersion="9"*

*android:targetSdkVersion="15" />*

*<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS" />*

*<uses-permission android:name="android.permission.RECORD\_AUDIO" />*

*<uses-permission android:name="android.permission.VIBRATE" />*

*<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />*

*<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />*

*<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />*

*<uses-permission android:name="android.permission.INTERNET" />*

*<uses-permission android:name="android.permission.CALL\_PHONE" />*

*<uses-permission android:name="android.permission.WAKE\_LOCK" />*

*<uses-permission android:name="android.permission.DISABLE\_KEYGUARD" />*

*<uses-permission android:name="android.permission.CAMERA" />*

*<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />*

*<uses-permission android:name="android.permission.WRITE\_SETTINGS" />*

*<uses-permission android:name="android.permission.READ\_CONTACTS"/>*

*<uses-permission android:name="android.permission.VIBRATE" />*

*<application*

*android:name=".MainApplication"*

*android:icon="@drawable/ic\_launcher"*

*android:label="@string/app\_name"*

*android:theme="@android:style/Theme.NoTitleBar" >*

//YzxIMCoreService是云之讯IM的核心服务，必须添加，通过Application调用启动服务

<service android:name="com.yzxtcp.service.YzxIMCoreService" ></service>

// AlarmReceiver和MsgBackReceiver是维持TCP长连接心跳的广播

<receiver android:name="com.yzxtcp.tools.tcp.receiver.AlarmReceiver" >

</receiver>

<receiver android:name="com.yzxtcp.tools.tcp.receiver.MsgBackReceiver" >

</receiver>

*</application>*

# 连接平台和注销

## 初始化IM服务

方法1：可以使用直接继承IMApplication类的方式，IMApplication类中已经包含了核心服务的初始化，代码如下：

public class MainApplication extends IMApplication {

@Override

public void onCreate() {

super.onCreate();

}

}

方法2：如果您的Application已经继承了其他第三方提供的Application，可以在Application中添加以下代码

public class MyApplication extends Application {

@Override

public void onCreate() {

super.onCreate();

UCSManager.init(this);//初始化核心服务

IMManager.getInstance(this);//必须要加上

}

}

## 连接平台

1. token方式登入（所有关于云之讯IM的操作都必须在登入成功后才能使用）

token是从AS服务器（用户服务器，您必须拥有自己的服务器）获取，token由AS服务器按照一定规则生成，token生成规则请看：<token说明见官网>

UCSManager.connect(String token, new ILoginListener() {

@Override

public void onLogin(UcsReason arg0) {

if(reason.getReason()==0){

//登入成功

例如登入成功后跳转到会话界面，显示获取的所有会话列表

}else{

//登入失败

例如登入失败后提示用户登入失败

}

}

});

1. 账户密码方式登入（仅限老平台用户继续使用，新接入者无需关心这种登入方式）

UCSManager.connent(String sid, String sidPwd，String clientId，String clientPwd， new ILoginListener() {

@Override

public void onLogin(UcsReason arg0) {

if(reason.getReason()==0){

//登入成功

//例如登入成功后跳转到会话界面，显示获取的所有会话列表

}else{

//登入失败

//例如登入失败后提示用户登入失败

}

}

});

## 注销

如果需要断开SDK与云平台连接(比如说切换账号时)，可以调用注销接口。如果需要发送消息或者接收消息等操作时必须重新调用UCSManager.content登入。注销函数如下：

UCSManager.disconnect();//注销函数

# 基本功能集成

## 获取IMManager类实例

IM各种功能实现方法，都可以通过imManager调用，比如获取所有会话列表、创建讨论组得到所有未读消息等，因此需要先获取imManager，获取方式如下：

IMManager imManager = IMManager.getInstance(this); //获得IMManager类实例

## 会话操作

接下里将介绍关于会话的一些基本操作，与某人聊天，一个讨论组或者群组都称之为一个会话。

1、设置会话回调监听器

首先Activity需要实现 IConversationListener，代码示例如下：

public class ConversationFragment extends Fragment implements IConversationListener

会话界面启动时在onCreate函数中就要首先添加会话回调监听器，这样才可以及时收到创建会话、删除会话和更新会话的回调接口。

IMManager.getInstance(this).setConversationListener(*mContext*);

2、 会话回调接口

当第一次与某人聊天时或者被加入到某个讨论组时，只要收到一条消息或者发送一 条消息，都会创建一个新的会话，会收到创建会话接口的回调。

void onCreateConversation(ConversationInfo cinfo) {

//把cinfo添加到会话列表中，更新界面

}

当你退出某个讨论组时，会删除这个讨论组会话，同时收到删除这个会话的回调。

void onDeleteConversation(ConversationInfo cinfo) {

//把cinfo从会话列表中移除，更新界面

}

当某个会话中收到消息时，同时收到更新这个会话的回调，比如这个消息的未读消息数，首条消息内容的变化，在回调中需要处理后更新会话界面。

void onUpdateConversation(ConversationInfo cinfoSrc) {

// 通过比对会话的targetId （单聊时是对方账号，讨论组或者群组时是讨论组或者群组账

//号）是否相等找到要被更新的会话cinfoDest，然后从会话列表中移除cinfoDest，根据收到 //的 cinfoSrc配置cinfoDest中的属性，然后重新加入到会话列表中，可以参考下面的函数：

updataCinfo(cinfoSrc, cinfoDest);

}

private void updataCinfo(ConversationInfo cinfoSrc,ConversationInfo cinfoDest){

conversationLists.remove(cinfoDest);

cinfoDest.setDraftMsg(cinfoSrc.getDraftMsg());

cinfoDest.setLastTime(cinfoSrc.getLastTime());

if(cinfoSrc.getCategoryId()!=CategoryId.GROUP){

cinfoDest.setConversationTitle(cinfoSrc.getConversationTitle());

}

if(cinfoDest.getIsTop()){

conversationLists.add(0, cinfoDest);

}else{

conversationLists.add(topNum, cinfoDest);

}

//更新会话列表

}

3、获取所有会话列表

进入会话界面，如果需要显示所有会话列表，可以调用下面API函数获取。

List<ConversationInfo> conversationLists = new ArrayList<ConversationInfo>();//定义会话列表

conversationLists=IMManager.getInstance(mContext).getConversationList(); //获取会话列表

4、会话类在消息界面的应用

从会话界面跳转到消息界面时，可以通过intent把这个会话的类conversationinfo传递进来，示例代码如下：

Intent intent = new Intent(getActivity(), IMMessageActivity.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent.putExtra("conversation", conversationinfo);

startActivity(intent);

在消息界面中可以通过conversationinfo类中的方法进行下列操作：

//需要获取最近的20条聊天记录

List<ChatMessage> currentMsgList = new ArrayList<ChatMessage>();

currentMsgList = conversationinfo.getLastestMessages(0, 20);

//清除会话未读消息

conversationinfo.clearMessagesUnreadStatus();

//清除会话聊天记录

conversationinfo.clearMessages();

//设置会话是否置顶

conversationinfo.setIsTop(Boolean isTop);

## 消息操作

1. 设置消息回调监听器

首先Activity需要实现 MessageListener，代码示例如下：

IMManager.getInstance(this).setSendMsgListener(*mContext*);

public class IMMessageActivity extends implements MessageListener

消息界面启动时在onCreate函数中就要首先添加消息回调监听器，这样才可以及时接受到消息接口的回调。

1. 消息回调接口

发送消息后会收到发送消息的回调，回调接口中收到的消息对象就是你之前发送的消息对象。

void onSendMsgRespone(ChatMessage message) {

//因为消息状态会改变，因此需要更新界面

}

收到一条或者多条消息后会在消息界面收到接收消息的回调。

void onReceiveMessage(List messages) {

//messages 保存的对象就是List <ChatMessage>，可以强转到 List <ChatMessage>

//把接收到的消息添加到消息列表，更新界面

}

调用IMManager中的方法downloadAttached（方法使用见SDK手册）后会收到下载回调，当下载附件或者大图片时候使用此方法和接收下面的回调接口。

void onDownloadAttachedProgress(String msgId, String filePaht, int sizeProgrss, int currentProgress) {

//获得文件下载进度，更新界面

}

1. 创建消息对象

消息对象包括单聊、群组和讨论组，他们都继承消息对象ChatMessage，下面是分别创建单聊、群组和讨论组的方法。

ChatMessage msg = null;

msg = new SingleChat();//创建单聊消息

msg = new GroupChat();//创建群组消息

msg = new DiscussionChat();//创建讨论组消息

1. 发送文本消息

ChatMessage msg = null;

msg = new SingleChat();//创建单聊消息

msg.setTargetId(对方账号);

msg.setSenderId(自己账号);

msg.setMsgType(MSGTYPE.MSG\_DATA\_TEXT);//设置消息类型为文本

msg.setContent(text); //设置消息内容

if(IMManager.getInstance(mContext).sendmessage(msg)){ //发送消息成功返回true

//发送成功后把消息添加到消息列表中，收到消息发送回调后刷新界面

currentMsgList.add(msg);

}

1. 发送图片消息

ChatMessage msg = null;

msg = new GroupChat ();//创建群聊消息

msg.setTargetId(对方账号);

msg.setSenderId(自己账号);

msg.setMsgType(MSGTYPE.MSG\_DATA\_IMAGE); //设置消息类型为图片

msg.setContent(path); //设置缩量图片路径

msg.setPath(path); //设置图片路径

IMManager.getInstance(mContext).sendmessage(msg) { //发送消息成功返回true

//发送成功后把消息添加到消息列表中，收到消息发送回调后刷新界面

currentMsgList.add(msg);

}

1. 发送语音消息

ChatMessage msg = null;

msg = new DiscussionChat ();//创建讨论组消息

msg.setTargetId(对方账号);

msg.setSenderId(自己账号);

msg.setMsgType(MSGTYPE.MSG\_DATA\_VOICE); //设置消息类型为语音

msg.setPath(path);//设置语音路径

msg.setContent(time);//设置语音时间

IMManager.getInstance(mContext).sendmessage(msg) { //发送消息成功返回true

//发送成功后把消息添加到消息列表中，收到消息发送回调后刷新界面

currentMsgList.add(msg);

}

1. 录音

云之讯SDK有提供专门的录音和播放接口(具体操作见SDK接口文档)，这里介绍录音接口，开发者可以选择使用我们提供的接口，下面是录音接口的使用方法：

IMManager.getInstance(mContext).startVoiceRecord(path, new RecordListener() {

@Override

public void onFinishedRecordingVoice(int time) {

if(time<1){

//提示用户时间小于1秒，无法发送，小于1秒请不要发送消息

return;

}

发送语音消息

}

@Override

public void onFinishedPlayingVoice() {

}

}

});

## 讨论组操作

1. 进入对讨论组操作的界面时（比如讨论组信息界面,创建讨论组界面），在onCreate时首先要添加讨论组监听接口，调用下面方法实现：

IMManager.getInstance(this).setDiscussionGroup (*mContext*);

1. 讨论组回调接口

void onCreateDiscussion(UcsReason reason, DiscussionInfo dInfo) {

if(reason.getReason()==0){

//创建成功

ConversationInfo info = IMManager.getInstance(this). getConversation(discussionInfo.getDiscussionId());

if(null != info){

intent.putExtra("conversation", info);

startActivity(intent);

finish();

}else{

CustomLog.e("获得讨论组会话为空");

}

}else{

//创建失败

}

}

void onDiscussionAddMember(UcsReason reason) {

if(reason.getReason()==0){

//加人成功

}else{

//加人失败

}

}

void onDiscussionDelMember(UcsReason reason) {

if(reason.getReason()==0){

//踢人成功

}else{

//踢人失败

}

}

void onQuiteDiscussion(UcsReason reason) {

if(reason.getReason()==0){

//退出成功

}else{

//退出失败

}

}

1. 创建讨论组

参数:

discussionID - 讨论组名字

memberList - 将添加的成员列表 IMManager.getInstance(mContext).createDiscussionGroup(java.lang.String discussionName,

java.util.List<java.lang.String> memberList);

1. 添加讨论组成员

参数:

discussionID - 讨论组ID

memberList - 将添加的成员列表 IMManager.getInstance(mContext).addDiscussionGroupMember(java.lang.String discussionID,

java.util.List<java.lang.String> memberList)

1. 删除讨论组成员

参数:

discussionID - 讨论组ID

memberList - 将添加的成员列表 IMManager.getInstance(mContext).delDiscussionGroupMember(java.lang.String discussionID,

java.util.List<java.lang.String> memberList)

1. 退出讨论组

参数:

discussionID - 讨论组ID

IMManager.getInstance(mContext).quitDiscussionGroup (java.lang.String discussionID)

1. 修改讨论组名称

参数:

discussionID - 讨论组ID

newname – 新的讨论组名称

imManager.modifyDiscussionTitle(java.lang.String discussionID, java.lang.String newname);

## 群组操作

SDK仅仅是提供群组消息的收发功能（SDK提供了讨论组的管理功能），群组的管理功能由用户AS服务器处理，群组消息发送参考3.3消息操作。

ChatMessage msg = new GroupChat()是创建群组消息对象。

void onReceiveMessage(List<ChatMessage> messages)是接收消息回调。

判断message. categoryId== CategoryId.GROUP表示消息是群组消息。

# 对象类属性介绍

## 会话类

private String targetId; //单聊id,群组id,讨论组id

private String conversationTitle; //会话标题

private int categoryId; //会话分类：1个人，2群组，3讨论组

private String draftMsg=""; //草稿或者是最近的消息

private boolean isTop; //是否置顶

private long lastTime; //最后消息时间

private long topTime; //置顶多久，默认是NULL

private int msgUnRead;

## 消息类

private int msgid; //消息ID

private int targetId; //接收用户名

private int senderId;//发送方用户名

private int categoryId;// 消息分类 1单聊，2群聊， 3，讨论组

private boolean isFromMyself;//消息方向 发送或者接收

private long sendTime;// 发送时间

private long receiveTime;//接收时间

private int msgType;//消息类型 1，文本，2图片，3语音

private String content;//消息类容：文本代表文字内容 图片代表小图路径 语音代表录音长度

private int readStatus;// 消息是否已读状态

private int sendStatus;//消息发送状态 1发送中，2，发送成功，3，发送失败

private String path; //大图片 语音 视频 的存储路径

private String parentID; //群组或讨论组 ID号

public final static int MSG\_STATUS\_INPROCESS = 1; //初始状态

public final static int MSG\_STATUS\_SUCCESS = 2; //发送成功状态

public final static int MSG\_STATUS\_FAIL = 3; //发送失败状态

public final static int MSG\_STATUS\_READED = 4; //已读状态

public final static int MSG\_STATUS\_UNREAD = 5; //未读状态

public final static int MSG\_STATUS\_RETRY = 6; //重发状态

## 讨论组信息类

private String discussionId; //讨论组ID

private String discussionName; //讨论组名字

private int categoryId; //聊天类型 1,单聊，2,群组，3,讨论组

private int memberCount; //讨论组成员数量

private String ownerId; //群组ID号

private String discussionMembers; //包含成员id，名字，头像

private String disscussionSettings; //讨论组设置信息，json

private long createTime; //讨论组创建时间

## 群组信息类

private String groupId; //群ID

private String groupName; //群名字

private int categoryId; //聊天类型 1,单聊，2,群组，3,讨论组

private int updateTime; //创建时间

## 返回值类

private int reason; //返回消息错误码

private String msg; //返回原因

## 会话类型枚举类

public enum CategoryId {

NONE,// 未知会话类型

PERSONAL, //单聊会话

GROUP, //群聊会话

DISCUSSION;// 讨论组会话

}

## 消息类型枚举类

public enum MSGTYPE {

MSG\_DATA\_NONE,// 未知消息类型

MSG\_DATA\_TEXT,// 文本消息

MSG\_DATA\_IMAGE,// 图片消息

MSG\_DATA\_VOICE,// 语音消息

MSG\_DATA\_VIDEO,// 视频消息

MSG\_DATA\_SYSTEM;// 系统消息

}