

# 客服 IMSDK 开发指南(Android)(4.9.9)

#客服 IMSDK 开发指南 (Android)

## 概述

快智乐服 Q 平台云客服平台访客端 Android SDK，可由对接企业移动端开发人员通过集成开发实现 App 内访客端的接入。

我们提供了较为完善的完全开源的 Demo 以及 Demo ui 代码。对接之前，对接企业的负责人或已开通坐席平台的管理员可以下载 Demo 提供给研发进行运行体验，提前了解 SDK 能力便于更快速的进行 SDK 集成和开发工作。

Demo ui 提供了一套基础的会话开源的 UI 示例，如不满足您的样式需求需要您对 Demo ui 样式进行自定义修改，请详细阅读并了解 Demo 代码后进行自定义 ui。

## SDK 说明

### 1、sdk 文件包含内容：

- 对接说明文档
- 示例 demo（AndroidSupport 和 AndroidX 版本）

### 2、sdk 运行环境及使用依赖库：

- minSdkVersion：19（使用视频最低 21）
- targetSdkVersion：33
- gradle 插件版本：最低 build:gradle:3.0.0
- gradle 版本：最低 gradle-4.1

依赖库：

- com.google.android:flexbox:1.0.0

- com.github.bumptech.glide:glide:4.9.0

```
-| com.github.hzlif:android-imkf-sdk:4.9.8

|- com.effective.android:panelSwitchHelper:1.3.13

|- com.github.7moor-tech:7moor_baseLib_sdk

|-org.greenrobot:eventbus:3.2.0

|-com.google.code.gson:gson:2.8.5

|-com.qiniu:qiniu-android-sdk:8.4.0

|-com.j256.ormlite:ormlite-core:5.1

|-com.j256.ormlite:ormlite-android:5.1
```

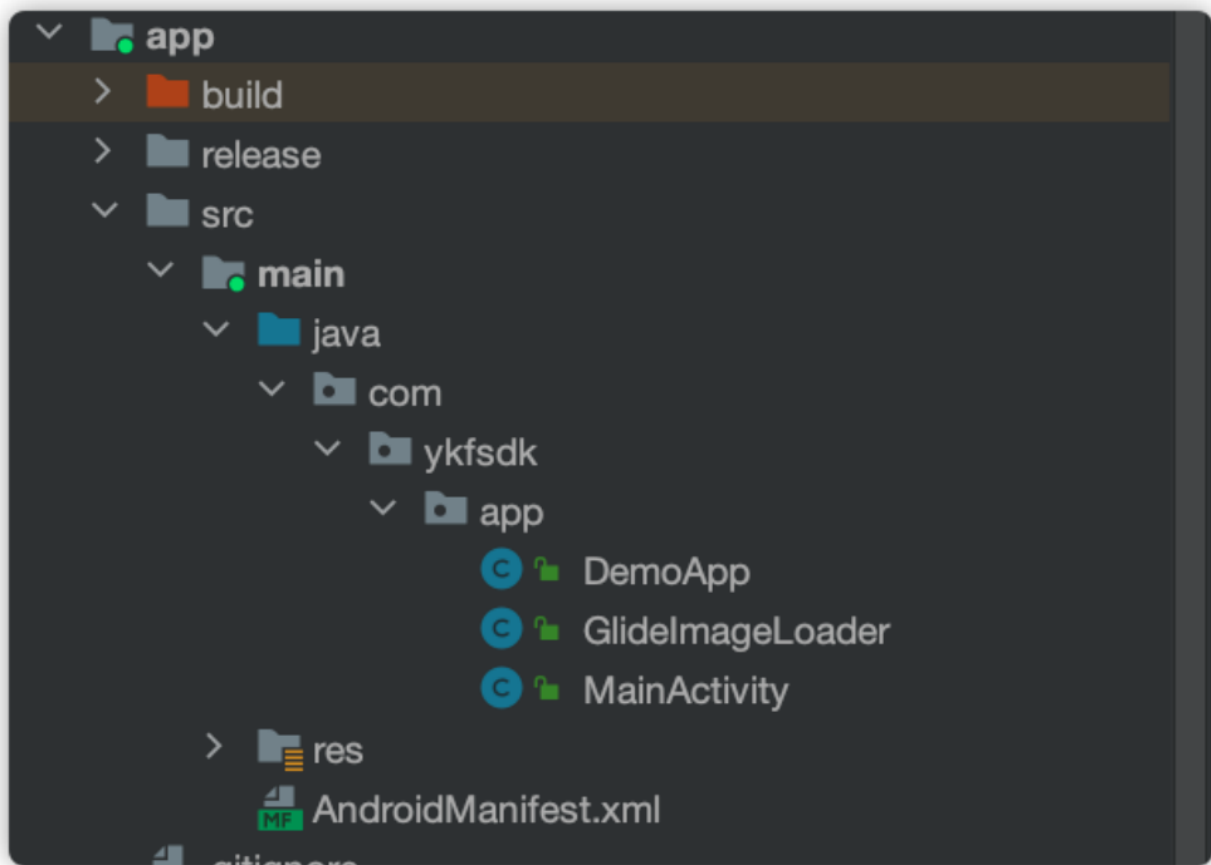
### 3、accessId 获取方法

登录坐席 pc 网站，点击左侧最下面的设置，选择左侧面板中的渠道设置，选择移动客服 APP，点击添加按钮，即可创建一个应用的接入号。在创建好的应用点击配置说明可查看对应的 accessId,用此 ID 即可初始化客服 SDK。



### 4、运行 DEMO 项目：

- 该项目为 android studio 项目，请用 android studio 进行打开
- 项目打开之后填写参数后可直接运行(参数修改文件为 MainActivity);



```
helper.initSdkChat("请填写后台获取的 accessId", "访客昵称", "访客 Id");
```

(填写获取的 accessId 后可直接运行 demo,查看效果)

**提醒：**accessId + userId +设备号 是访客的唯一标识，尽量保证 userId 的唯一性

## 三、开始集成

### 1、无自定义 UI 需求-导入依赖

如果 demo 提供的 ui 样式能够满足需求，并且无需对 ui 以及页面逻辑进行任何改动，可以直接在项目中添加依赖：

1：Android Support 项目

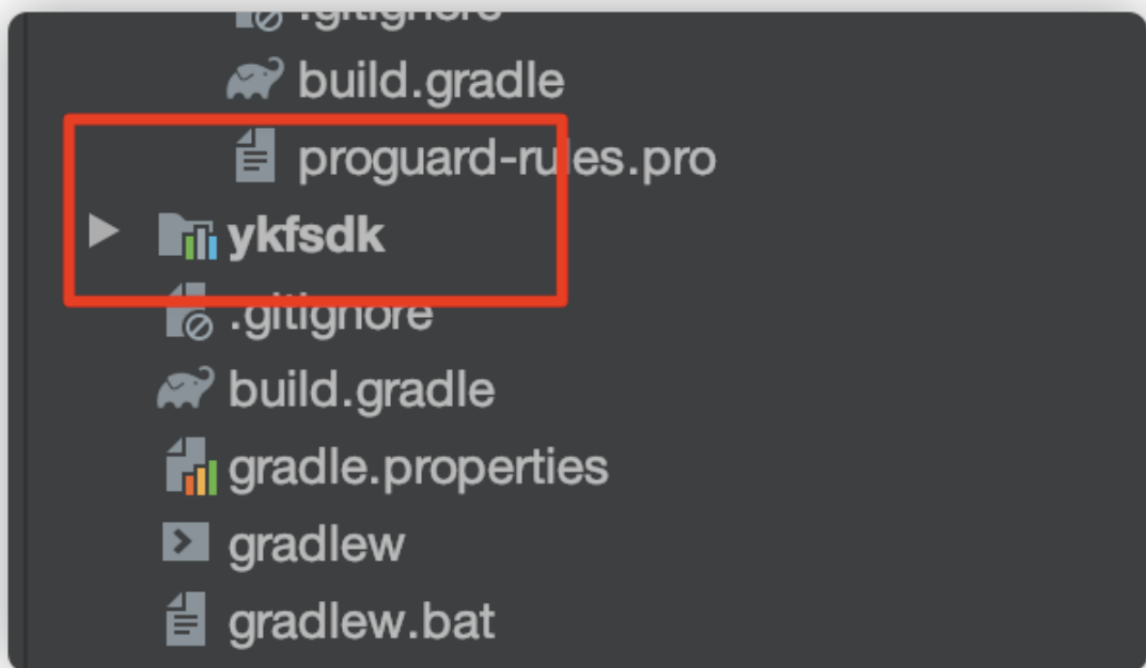
api 'com.github.kzlf:android-ykf-sdk:version' version 为 SDK 的版本, 比如当前版本 4.9.8

2: Android X 项目

api 'com.github.kzlf:androidx-ykf-sdk:version'

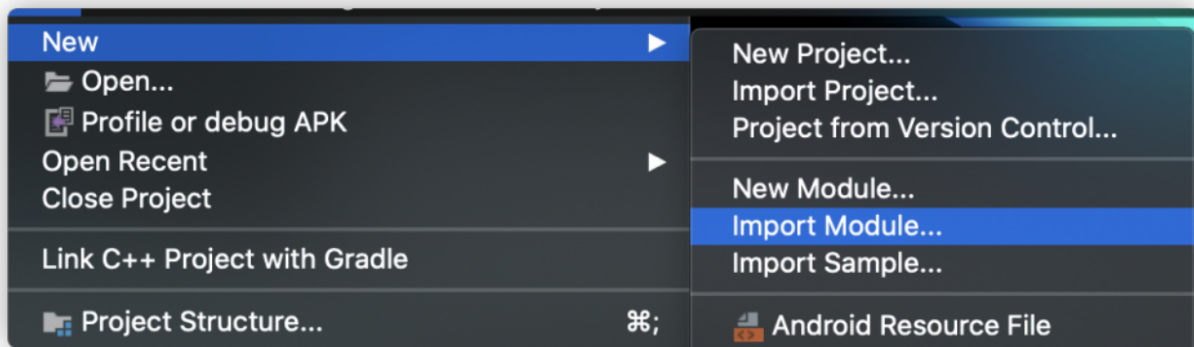
依赖版本号具体可以参考下载压缩包中的 kfsdk\_moudle\_AAR 项目, 使用其配套的远程版本号。

## 2、自定义 UI 以及页面自定义调整-导入 ykfsdk Moudle 形式

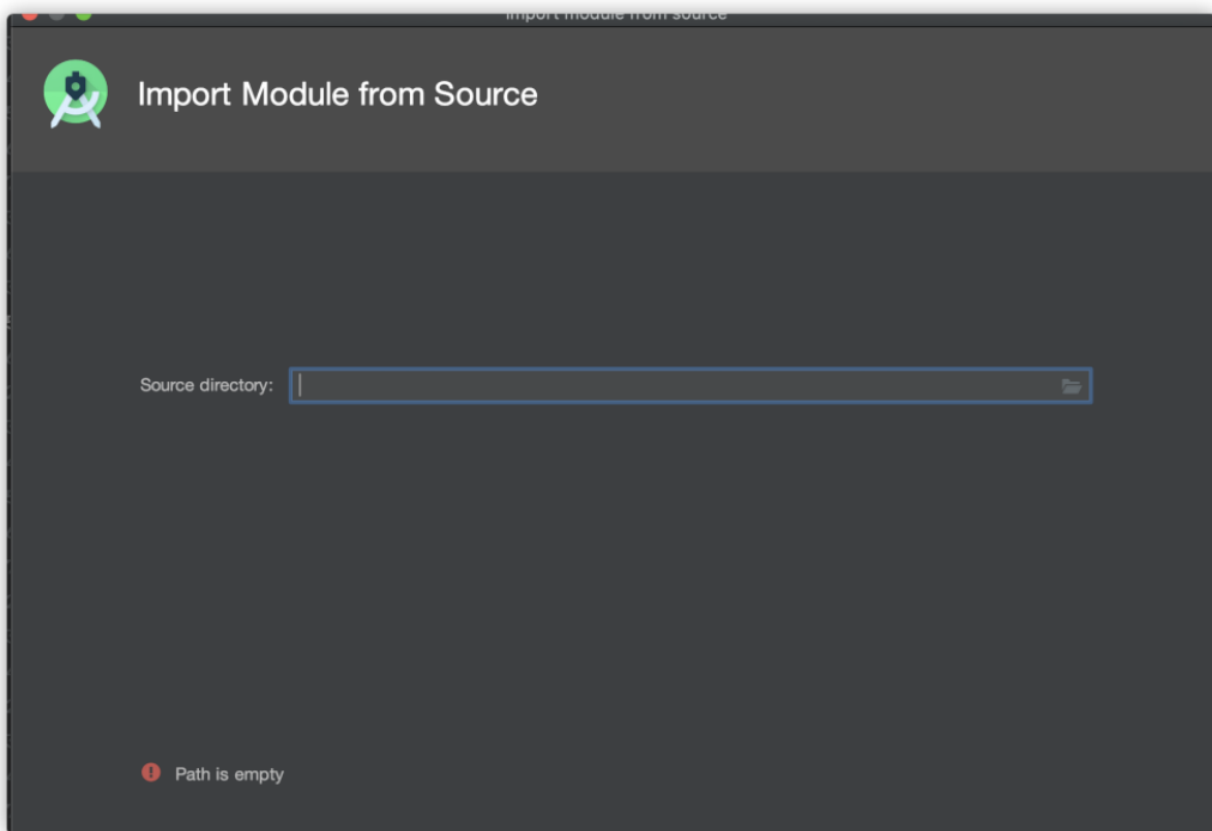


具体步骤

1:使用 Android studio 导入 Module



2:在窗口中选择到 ykfsdk 的 modlue 中，进行下一步导入。



3:在自己项目 app 目录下的 build.gradle 中增加:

```
api project(path: ':ykfsdk')
```

4:重新 build 你的项目。

### 3、项目 **app/build.gradle** 配置

1:配置 sdk 内部使用到的 fileprovider;路径

```
defaultConfig {;
```

```
.....
```

```
manifestPlaceholders.put("IMKF_PROVIDER_PATH", ".fileprovider");
```

```
.....
```

```
}
```

参考 demo 中 app/build.gradle 配置

2:配置 so 库环境

根据实际项目情况按需配置

so 库支持: 'armeabi', 'armeabi-v7a', 'arm64-v8a', 'x86', 'x86\_64'

```
defaultConfig {;
```

```
.....
```

```
ndk {;
```

```
abiFilters 'armeabi-v7a', 'arm64-v8a', 'x86';
```

```
}
```

```
.....
```

```
}
```

参考 demo 中 app/build.gradle 配置

## 4、使用音视频会话功能

如果需求上需要使用到音视频会话功能，那么在完成以上依赖导入后再添加如下依赖并完成相关代码：

音视频仓库：[https://central.sonatype.com/artifact/com.github.7moor-tech/sdk\\_video\\_jitsi/1.0.8/overview](https://central.sonatype.com/artifact/com.github.7moor-tech/sdk_video_jitsi/1.0.8/overview)

1：视频依赖库：api 'com.github.7moor-tech:sdk\_video\_jitsi:1.0.8'

2：项目根目录添加仓库：

```
allprojects {
    repositories {
        .....
        maven { url "https://raw.githubusercontent.com/7moor-tech/video-call-sdk/master" }
        .....
    }
}
```

3：使用音视频功能项目必须为 Androidx 项目，不支持 Android support 项目。

4：使用音视频功能项目的 minSdkVersion 最低为 21。

5：对 app gradle 开启 java8 支持。

```
android {
    .....
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    .....
}
```

6：音视频 sdk 混淆规则处理，详细请查看文档中的"混淆配置"部分。

## 5、备注

- 如果没有特定场景需求，本 demo 示例基本满足大部分需求，用户将 demo 导入自己项目后，完成相关的初始化配置，输入 accessid 后并确认环境就可完成配置。
- **特别注意** (*NewMsgReceiver* 这个类的路径必须是 *com.m7.imkfsdk.receiver*，否则会导致无法及时收到消息)。注意：sdk 初始化可以直接参考 demo 中 *DemoApp,MainActivity* 的代码。
- 如果有关于头像、商品链接、扩展信息字段请参考本文章中的延伸功能。
- 以下接口说明是针对特殊场景开发以及向开发人员说明接口功能。
- 音视频 SDK 遇到 video-call-sdk 依赖无法下载或下载困难，请查看网络是否可以正常访问 github，可以上网获取 github host 修改方案调整电脑的网络配置。

## 四、sdk 使用说明

### 开始使用 SDK

再此之前请确保 moudle 已成功导入并且 gradle 等配置已完成。

**1：在项目 Application 的 onCreate 中添加**

```
YKFUtils.init(this);
```

此方法不会涉及到权限获取可直接调用。

参考 demo 中的 DemoApp。



```

17 public class DemoApp extends Application {
18     @Override
19     public void onCreate() {
20         super.onCreate();
21         /**
22          * 1 (重要) :
23          * 请检查在 Application onCreate() 中的初始化
24          * 初始化YKFUtils : YKFUtils.init(this)
25          *
26          * 注意:如果使用三方crash监控等插件,请在初始化三方crash监控之前init本sdk,像下面demo这样的顺序.
27          */
28         YKFUtils.init(application: this);
29
30
31         //bugly(demo 演示用,可移除)
32         CrashReport.initCrashReport(getApplicationContext(), s: "33d895df1e", b: true);
33
34         //友盟(demo 演示用,可移除)
35         UMConfigure.init(context: this, s: "64184af1ba6a5259c41da93e", s1: "Demo", UMConfigure.DEVICE_TYPE_PHONE, s2: "");
36         UMConfigure.setLogEnabled(true);
37
38     }
39 }

```

注意:如果遇到三方监控无法收集错误的情况, 请参考文档常见问题章节。

## 2: 创建 KfStartHelper 对象

详细内容代码参考 demo 中的 MainActivity 部分

```
x0; KfStartHelper helper = KfStartHelper.getInstance();
```

调用配置要使用的服务环境:

目前提供两种方法进行配置

```
1: RequestUrl.setRequestBasic();
```

*腾讯云, 阿里云用户只使用 setRequestBasic 即可*

```
RequestUrl.ALIYUN_REQUEST;//阿里云环境
```

```
xa0; xa0; xa0; RequestUrl.TENCENT_REQUEST;//腾讯云环境
```

```
RequestUrl.HUAWEI_REQUEST;//华为云环境 x0;
```

2: RequestUrl.setRequestUrl(); 注意: 此方法仅适用于私有云环境

`RequestUrl.setRequestUrl()(String http)`

参数说明 http: 使用的 Http 服务

*注意: `setRequestBasic` 与 `setRequestUrl` 为二选一调用, 腾讯云、阿里云、华为云只使用 `setRequestBasic` 即可, 私有云用户只使用 `setRequestUrl` 即可。*

3: `RequestUrl.setFileUrl ()` ;配置私有云文件服务器

*注意: 此方法仅适用于私有云环境, 腾讯云、阿里云、华为云客户无须调用。*

```
/* 要在 helper.initSdkChat()之前设置
 * fileHttp"完整的服务地址",
 * fileDomains: {"只需填写中间域名以及端口部分"},
 * isHttps: 是否为 https
 */
```

```
RequestUrl.setFileUrl( "https://im-fileserver:8000/",
    new String[]{"im-fileserver:8000"},true)
```

### 3:配置 SDK 使用的图片加载器

```
helper.setImageLoader(new GlideImageLoader());x0;
```

配置 ykf sdk 中 ui 部分使用的图片加载框架, demo app 提供了 Glide(4.9.0 版本)的 `GlideImageLoader` 加载器实现, 可以根据自己的项目需要进行开发替换。

自行实现可以实现 `IMoorImageLoaderx0`接口, 参考 `GlideImageLoader` 的处理方式, 。

### 4:开始进入 SDK

```
helper.initSdkChat("", "", "");
```

其中参数说明:

`String accessId`, 接入号,必填项.

`String username`, 用户名, 用来在后台显示.(不能为 null 或空字符串)

`String userId`, 用户 id, 用来标识用户(建议唯一).(不能为 null 或空字符串)

## SDK 初始化的接口监听

```
MChatManager.getInstance().setOnInitListener(new OnInitListener() {  
    @Override  
    public void onInitSuccess() {  
        LogUtil.d("MobileApplication", "sdk 初始化成功");  
    }  
    @Override  
    public void onInitFailed(int code) {  
        LogUtil.d("MobileApplication", "sdk 初始化失败");  
    }  
});
```

注意：该回调接口只是用来判断 SDK 是否初始化成功了，注意：只有成功了之后才可以使用 IM 相关功能,该返回接口是在主线程中，可以直接操作 UI。

## 配置是否使用前台服务

默认不使用前台服务，使用前台服务可以提高连接的稳定性，可以选择开启。

(需要在 initSdkChat 方法执行前调用)

```
//配置是否使用前台服务，默认 false 不使用前台服务，true 开启前台服务  
IMChatManager.getInstance().setUSE_ForegroundService();  
helper.initSdkChat("", "", "");
```

注意：需要在 initSdkChat()方法之前调用。

获取配置日程管理接口。

开始会话前先得获取是否配置日程，代码如下：

具体使用详见 demo

```
IMChatManager.getInstance().getWebchatScheduleConfig(InfoDao.getInstance().getConnection  
Id(), new GetGlobeConfigListen() {  
    @Override  
    public void getSchedule(ScheduleConfig sc) {  
  
    }  
    @Override  
    public void getPeers() {
```

```
    }  
});
```

## 获取技能组接口

开始会话前先得获取后台配置的技能组，代码如下：

```
IMChatManager.getInstance().getPeers(new GetPeersListener() {  
    @Override  
    public void onSuccess(List<Peer> peers) {  
    }  
    @Override  
    public void onFailed() {  
    }  
});
```

onSuccess 接口中返回的即是技能组的数据列表

## 会话开始接口

当开始一次新会话时，需请求该接口通知服务器，并将需要联系的技能组 id 传进来，代码如下：

```
IMChatManager.getInstance().beginSession(peerId, new OnSessionBeginListener() {  
    @Override  
    public void onSuccess() {  
    }  
    @Override  
    public void onFailed() {  
    }  
});
```

该接口的回调中

onSuccess 表示会话开始成功了

onFailed 表示接口请求失败，具体请参考提供的 demo。

## 广播接收器

调用过该接口后，需在聊天界面中注册广播来接收当前是机器人还是人工客服的状态，进行界面的相应变化。广播接收器代码如下：

```

class KeFuStatusReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if(IMChatManager.ROBOT_ACTION.equals(action)) {
            //当前是机器人
            handler.sendMessage(0x111);
        } else if(IMChatManager.ONLINE_ACTION.equals(action)) {
            //当前客服在线
            handler.sendMessage(0x222);
        } else if(IMChatManager.OFFLINE_ACTION.equals(action)) {
            //当前客服不在线
            handler.sendMessage(0x333);
        }
    }
}

```

具体请参看 demo;

## 获取当前会话接入状态

当前会话接入状态是人工还是机器人或者是排队中

IMChatManager.getInstance().getYkfChatStatusEnum()

## 获取后台满意度是否开启

获取后台满意度是否开启的配置

在调用完 beginSession 成功后，可以通过.

IMChatManager.getInstance().isInvestigateOn()

方法来获取满意度评价是否开启，来隐藏或显示评价按钮。

## 转人工客服

当服务器配置了机器人聊天后，通过转人工接口可以与人工客服进行聊天，代码如下：

```

/**
 * 转人工服务
 * @param peerId 技能组 id
 * @param bot2human_reason 转人工按钮点击传 type 字符串 "14", 消息体中的点击传
type 字符串 "13"

```

```

        */
IMChatManager.getInstance().convertManual(String peerId,
                                           String bot2human_reasonnew ,
                                           OnConvertManualListener() {

@Override
public void onLine() {
//有客服在线，隐藏转人工按钮
Toast.makeText(ChatActivity.this, "转人工服务成功", Toast.LENGTH_SHORT).show();
}

@Override
public void offLine() {
//当前没有客服在线，弹出离线留言框
}
});

```

成功之后，则可以与人工客服进行聊天。

## 提交离线留言

当客服不在线时，可以提交离线留言，代码如下：

### 1：技能组留言

参考 `OfflineMessageActivity`;

### 2：日程留言

参考 `ScheduleOfflineMessageActivity`;

## 消息实体

界面显示时会用到消息的一些属性进行不同的显示，下面将消息中的具体属性展示如下：

消息实体类为 `FromToMessage`.

属性	说明
userType	用户发送消息的类型：发送的消息为”0”，接收的消息为”1”

msgType	消息类型： 分别为文本消息， 录音消息， 图片消息, xa0;知识库文本.....
when	消息发送或接收的时间
message	消息的文本内容
sendState	消息发送的状态
recordTime	若为录音消息， 表示录音时长
filePath	录音文件或图片文件在本地的路径

更多详细请查看 demo 中的 FromToMessage

## 拼装消息

FromToMessage0;为 sdk 中使用的消息体对象， 使用 ormlite 进行数据库存储。

拼装消息： 通过 sdk 提供的 IMMessage0;类可以创建各种类型的消息。

可以通过 ChatActivity 中对此类的使用进行参考。

数据库插入消息： 使用数据库 dao 方法可以进行插入。

```
MessageDao.getInstance().insertSendMsgsToDaox0;()
```

## 发送消息

使用如下代码:

```
IMChat.getInstance().sendMessage(fromToMessage, new ChatListener() {
@Override
public void onSuccess() {
}
@Override
public void onFailed() {
}
@Override
public void onProgress() {
}
});
```

参数说明:

FromToMessage fromToMessage, 要发送的消息

ChatListener ,消息发送的接口监听，发送成功，失败或正在发送，该回调接口中可以直接进行界面的操作。发送的消息存到了本地数据库中，具体可参看提供的 demo

## 重发消息

当有时候发送失败后，需重新发送该条消息，代码如下：

```
IMChat.getInstance().reSendMessage(fromToMessage, new ChatListener() {  
    @Override  
    public void onSuccess() {  
    }  
    @Override  
    public void onFailed() {  
    }  
    @Override  
    public void onProgress() {  
    }  
});
```

## 接收新消息

需通过注册广播来获取新消息，首先需要全局注册的广播，在 AndroidManifest.xml 中代码为：

```
<receiver  
    android:name="com.m7.imkfsdk.receiver.NewMsgReceiver"  
    android:enabled="true"  
    >  
    <intent-filter android:priority="2147483647" >  
        <!--修改此 action 为自己定义的，该 action 必须和 SDK 初始化接口中所填的一样,举例  
如下-->  
        <action android:name="com.m7.demo.action" />  
    </intent-filter>  
</receiver>
```

请先修改此 action 的值，和 SDK 初始化时传入的值必须相同，否则接收不到新消息的广播，注意当接收到该广播后，消息已经保存到了本地的数据库中了。具体请查看 demo。

## 获取数据库中的消息列表

在界面上显示消息就得先从数据库中获得消息数据，代码如下：



```

/**
 * 从本地数据库中获取消息数据
 * 会根据 SP 中 YKFConstants.SAVE_MSG_TYPE 的取值判断过滤条件
 *
 * @param i 第几页的数据，默认一页 15 条，从第一页开始取 (i=1)
 * @return 消息数据列表
 */
List<FromToMessage>fromToMessages=IMChatManager.getInstance().getMessages(int i);

```

参数中的数字为取第几页的数据，用于下拉加载更多消息时使用,默认是一页 15 条消息数据。这样就获取到了数据库中的消息了，之后就可以在界面进行显示操作了。具体参考 demo 中。

### 配置查询消息列表的条件

sdk 目前提供的 demo 为通过客户传入的 userId + accessid 为条件查询数据库用作页面展示。

如果客户有需要其他的过滤条件需要取数据可以配置以下方法进行修改。

如果提供的几个条件不能满足需求客户可以使用自定义 sql 语句进行查询达到目的

```

/**
 * 配置 msg 数据库取出条件
 * 0:根据 userId + accessid 两个条件查询(默认)
 * 1:根据 peedId 查询 不适应于日程
 * 2:根据 accessid 查询
 * 3:根据 userId 查询
 * 注意:更改会保存到 SP 如果更换条件以后 需要重新配置
 */
public void setSaveMsgType(int type) {
    MoorSPUtils.getInstance().put(YKFConstants.SAVE_MSG_TYPE, type, true);
}

```

### 自定义 sql 语句数据库查询

以下几种情况可能实际需要使用 sql 语句进行数据库查询

- 1: 如果 sdk SAVE\_MSG\_TYPE 提供的默认 4 种形式不能满足消息列表的过滤
- 2: 需要自定义条件查询多个或者一个消息数据

```

/**
 * 使用自定义 sql 语句查询消息列表
 * 适用于客户根据自己的条件查询想要的消息
 * 注:返回的 msg 对象有可能因为版本关系字段不全,但是基本满足 msg 日常所需的字段,
 如遇到字段缺失可联系技术支持,后续升级提供支持
 *
 * @param sqlStr sql 语句
 * @return List<FromToMessage>
 */
MessageDao.getInstance().getMessagesBySql(String sqlStr)

```

### 更新一条消息数据到数据库中

若需要将消息的数据修改后保存到数据库中，代码如下：

```
IMChatManager.getInstance().updateMsgToDao(message);
```

参数为 FromToMessage message,修改数据后的消息。

### 获取评价列表数据

获取后台配置的评价列表数据，代码如下：

```
List<Investigate> investigates = IMChatManager.getInstance().getInvestigate();
```

返回的结果结果为：List investigates，即为评价实体的列表，Investigate 中的 name 属性为该评价的名称，可用来显示在界面上。

### 提交评价结果

代码如下：

```

IMChatManager.getInstance().submitInvestigate(investigate, new SubmitInvestigateListener() {
    @Override
    public void onSuccess() {
        Toast.makeText(InvestigateActivity.this, "评价提交成功",
        Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onFailed() {
        Toast.makeText(InvestigateActivity.this, "评价提交失败",Toast.LENGTH_SHORT).show();
    }
});

```

参数为：

Investigate investigate 评价实体

SubmitInvestigateListener 提交评价回调接口

注：该接口只有在与人工客服聊天过后评价才有意义。

### 客服主动发起的评价

客服在服务完客户后也可以主动发起评价的请求，客户端需要作出相应的响应，客户端在聊天界面注册相应的广播，接收到相应的事件会生成一个评价的消息，点击完评价之后该消息会被删除，生成评价消息的代码如下：

```
private void sendInvestigate() {  
    ArrayList<Investigate> investigates = (ArrayList<Investigate>)  
    IMChatManager.getInstance().getInvestigate();  
    IMMessage.createInvestigateMessage(investigates);  
    updateMessage();  
}
```

### 工号、姓名、头像推送

在接入聊天会话后，会把客服人员的工号、姓名和头像信息推送回来。聊天界面中通过接收 IMChatManager.USERINFO\_ACTION 的广播来获取对应数据。

### 访客无响应断开对话时长配置获取

通过 GlobalSet globalSet = GlobalSetDao.getInstance().getGlobalSet();来获取配置信息，里面的 break\_len 代表断开对话时长，break\_tips\_len 代表断开前提示时长，break\_tips 代表提示内容。

### 关闭 SDK

若需关闭 sdk 会话页面，则可以调用该接口，代码如下：

```
IMChatManager.getInstance().quitSdk();
```

注意：只是关闭访客这边 sdk 的会话页面，坐席侧会话依然存在，并不是关闭会话方法

### 开启关闭 log 打印

sdk 中包含帮助开发人员开发的数据 log,上线时可根据需求开关，代码如下：

```
//关闭 log
IMChatManager.getInstance().closeLog();
//打开 log
IMChatManager.getInstance().openLog();
```

## 获取 sdk 版本号

获取当前使用的 IMKFSdk 版本号

```
//获取 sdk 版本号
IMChatManager.getInstance().getSdkVersion();
```

## 混淆配置

因 sdk 中使用到七牛，eventbus，glide 等三方库，以下混淆代码除 ‘m7KF’ 片段外只针对 sdk 使用依赖库版本提供混淆示例，更多详细的混淆配置可根据三方依赖库官方文档配置。

如果配置了音视频依赖，那么视频部分的依赖一定要添加，如果没有配置那客户忽略这部分混淆

注意:shrinkResources 设置为 false，不然表情显示不正确

混淆代码：

```
-keepattributes *Annotation*
-keepattributes Signature
-keepattributes SourceFile,LineNumberTable
-keepattributes *DatabaseField*
-keepattributes *DatabaseTable*
-keepattributes *SerializedName*
-keep class **.R$* {*,}
-ignorewarnings
```

```
#####imkf 客服**必要配置
```

```
#####
```

```
#-----m7-----
```

```
-dontwarn com.m7.imkfsdk.**
-keep class com.m7.imkfsdk.** { *; }
-dontwarn com.moor.imkf.**
-keep class com.moor.imkf.** { *; }
```

```
#软键盘
```

```

-dontwarn com.effective.android.panel.**
-keep class com.effective.android.panel.** { *; }
#-----m7KF-----

#-----依赖库-----
#七牛
-dontwarn com.qiniu.**
-keep class com.qiniu.** { *; }
-keep class com.qiniu.** { public <init>(); }
# OkHttp3
-dontwarn okhttp3.**
-keep class okhttp3.** { *; }
# Okio
-dontwarn com.squareup.**
-dontwarn okio.**
-keep public class org.codehaus.* { *; }
-keep public class java.nio.* { *; }
#GSON
-dontwarn sun.misc.**
-keep class com.google.gson.examples.android.model.** { <fields>; }
-keep class * extends com.google.gson.TypeAdapter
-keep class * implements com.google.gson.TypeAdapterFactory
-keep class * implements com.google.gson.JsonSerializer
-keep class * implements com.google.gson.JsonDeserializer
-keepclassmembers,allowobfuscation class * {@com.google.gson.annotations.SerializedName
<fields>;}
#eventbus
-dontwarn org.greenrobot.eventbus.**
-keepclassmembers class * {
    @org.greenrobot.eventbus.Subscribe <methods>;
}
-keepclassmembers class ** {
    public void onEvent*(**);
}
-keep enum org.greenrobot.eventbus.ThreadMode { *; }
#glide
-dontwarn com.bumptech.glide.**
-keep public class * implements com.bumptech.glide.module.GlideModule
-keep class * extends com.bumptech.glide.module.AppGlideModule {
<init>(...);
}
-keep public enum com.bumptech.glide.load.ImageHeaderParser$** {
    **[] $VALUES;
    public *;
}
-keep class com.bumptech.glide.load.data.ParcelFileDescriptorRewinder$InternalRewinder {

```

```

    *** rewind();
}

#OrmLite, sqlcipher
-keep class com.j256.** { *; }
-keep class com.j256.**
-keepclassmembers class com.j256.**
-keep enum com.j256.**
-keepclassmembers enum com.j256.**
-keep interface com.j256.**
-keepclassmembers interface com.j256.**
#*****imkf 客服**必要配置
*****

```

```

#*****imkf 视频
*****
#如果不使用视频功能并且没有添加视频依赖 可以注释此部分混淆
#如果使用视频功能 一定要配置这部分混淆
# React Native

```

```

# Keep our interfaces so they can be used by other ProGuard rules.
# See http://sourceforge.net/p/proguard/bugs/466/
-keep,allowobfuscation @interface com.facebook.proguard.annotations.DoNotStrip
-keep,allowobfuscation @interface com.facebook.proguard.annotations.KeepGettersAndSetters
-keep,allowobfuscation @interface com.facebook.common.internal.DoNotStrip

```

```

# Do not strip any method/class that is annotated with @DoNotStrip
-keep @com.facebook.proguard.annotations.DoNotStrip class *
-keep @com.facebook.common.internal.DoNotStrip class *
-keepclassmembers class * {
    @com.facebook.proguard.annotations.DoNotStrip *;
    @com.facebook.common.internal.DoNotStrip *;
}

```

```

-keepclassmembers @com.facebook.proguard.annotations.KeepGettersAndSetters class * {

```

```
void set***);  
*** get*());  
}
```

```
-keep class * extends com.facebook.react.bridge.JavaScriptModule { *; }  
-keep class * extends com.facebook.react.bridge.NativeModule { *; }  
-keepclassmembers,includedescriptorclasses class * { native <methods>; }  
-keepclassmembers class * { @com.facebook.react.uimanager.UIProp <fields>; }  
-keepclassmembers class * { @com.facebook.react.uimanager.annotations.ReactProp  
<methods>; }  
-keepclassmembers class * { @com.facebook.react.uimanager.annotations.ReactPropGroup  
<methods>; }
```

```
-dontwarn com.facebook.react.**  
-keep,includedescriptorclasses class com.facebook.react.bridge.** { *; }  
# WebRTC
```

```
-keep class org.webrtc.** { *; }  
-dontwarn org.chromium.build.BuildHooksAndroid
```

```
# Jitsi Meet SDK
```

```
-keep class org.jitsi.meet.** { *; }  
-keep class org.jitsi.meet.sdk.** { *; }
```

```
# We added the following when we switched minifyEnabled on. Probably because we  
# ran the app and hit problems...
```

```
-keep class com.facebook.react.bridge.CatalystInstanceImpl { *; }  
-keep class com.facebook.react.bridge.ExecutorToken { *; }  
-keep class com.facebook.react.bridge.JavaScriptExecutor { *; }  
-keep class com.facebook.react.bridge.ModuleRegistryHolder { *; }  
-keep class com.facebook.react.bridge.ReadableType { *; }  
-keep class com.facebook.react.bridge.queue.NativeRunnable { *; }  
-keep class com.facebook.react.devsupport.** { *; }
```

```
-dontwarn com.facebook.react.devsupport.**  
-dontwarn com.google.appengine.**  
-dontwarn com.squareup.okhttp.**  
-dontwarn javax.servlet.**
```

```
# ^^ We added the above when we switched minifyEnabled on.
```

```
# Rule to avoid build errors related to SVGs.  
-keep public class com.horcrux.svg.** { *; }
```

# <https://github.com/facebook/fresco/issues/2638>

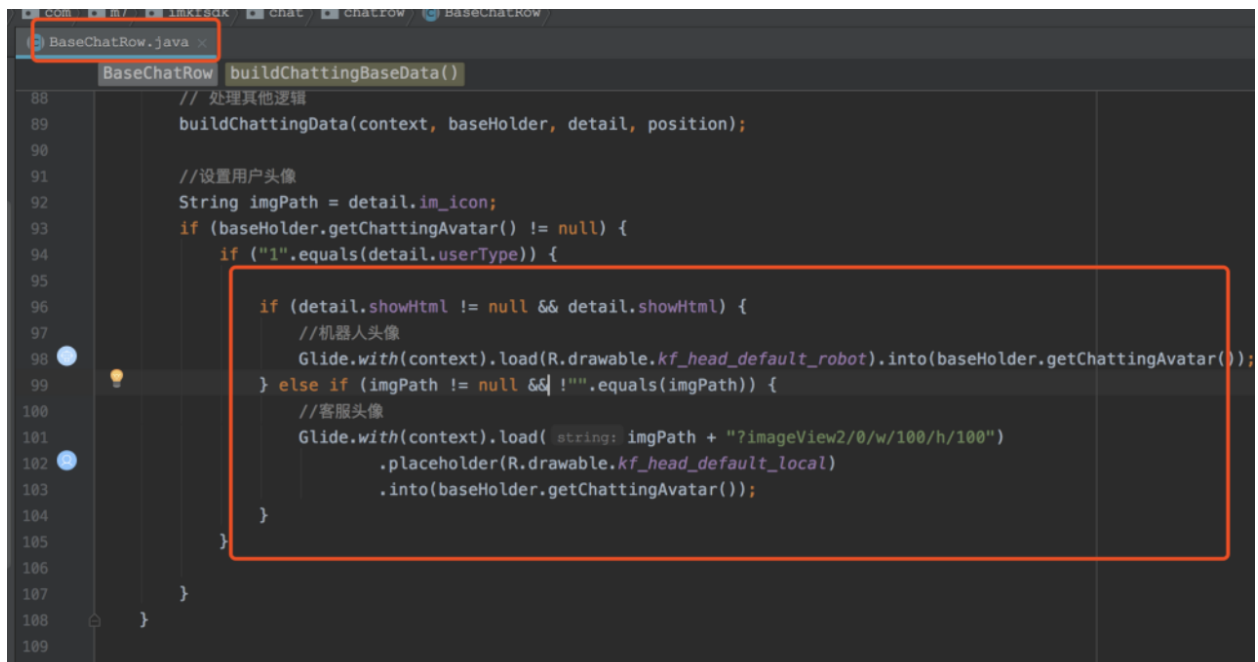
```
-keep public class com.facebook.imageutils.** {  
    public *;  
}
```

#####imkf 视频

#####

## 五、延伸功能

### 1、如需自定义用户头像和机器人头像



### 2、设置专属座席功能

sdk v3.8.0 提供了配置拓展信息的接口

- 版本号 $\geq 8.0$

在 demo 的初始化方法前提供了 setOtherParams()示例



如果需要设置 专属坐席 第一个参数传入坐席 id 即可

```
/**
 * 添加要使用的 拓展信息内容
 * @param agentId 专属坐席 id
 * @param otherObject: 添加拓展字段的数据
 * @param useUserLabels: 是否使用 user_labels 参数
 * @param user_labels: user_labels 字段数据
 */
public void setUserOtherParams(String agentId,JSONObject otherObject, boolean
useUserLabels, JSONObject user_labels)
```

- 版本小于 8.0

调用 beginSession 的重载方法，可传入 json 字符串，示例如下，需在 ChatActivity 中对原有的 beginSession 进行修改，不可单独调用

```
String otherParams = "";
JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put("agent", "8131");
} catch (JSONException e) {
    e.printStackTrace();
}
otherParams = jsonObject.toString();
IMChatManager.getInstance().beginSession(peerId, otherParams, new
OnSessionBeginListener() {})
```

### 3、设置扩展信息

sdk v3.7.0 提供了配置拓展信息接口

- 版本号 $\geq 7.0$

在 demo 的初始化方法前提供了 setOtherParams()示例

```
/**
 * 添加要使用的 拓展信息内容
 * @param agentId 专属坐席 id
 * @param otherObject: 添加拓展字段的数据
 * @param useUserLabels: 是否使用 user_labels 参数
 * @param user_labels: user_labels 字段数据
 */
```

```
public void setUserOtherParams(String agentId,JSONObject otherObject, boolean
useUserLabels, JSONObject user_labels)
```

在 ChatActivity 的 beginSession()和 beginScheduleSession()方法使用如下方法来获取字段  
String other\_Str = IMChatManager.getInstance().getUserOtherParams();

- 版本小于 7.0

调用 beginSession 的重载方法，可传入 json 字符串，示例如下，需在 ChatActivity 中对原有的 beginSession() 和 beginScheduleSession()进行修改，不用关注 beginsession 代码里执行了几次，只增加

```
//JSONObject j1 中可配置您的想要添加的扩展字段，
//key 与 value 根据需求要自定义。
JSONObject j1 = new JSONObject();
j1.put("name","测试扩展字段");
j1.put("other","前面字段可以自定义");

//将 j1 添加到 JSONObject j2 中，注意 j2 段代码必须是如下配置。
//不可修改与增加 j2 中的 key value。直接复制即可。
JSONObject j2 = new JSONObject();
j2.put("customField",URLEncoder.encode(j1.toString()));
j2.toString();
```

将 j2.toString() 得到的 String 分别添加到：

1:ChatActivity 中的 beginSession() 方法 添加：

```

private void beginSession(String peerId) {
    if (IMChatManager.getInstance().getAppContext() == null) {
        return;
    }
    // String otherParams = "";
    // JSONObject jsonObject = new JSONObject();
    // try {
    //     jsonObject.put("agent", "1002");
    // } catch (JSONException e) {
    //     e.printStackTrace();
    // }
    // otherParams = jsonObject.toString();

    // TODO: 2019/10/30 这里增加 扩展信息内容, 参照文档的传参方式,
    IMChatManager.userId = InfoDao.getInstance().getUserId();
    HttpManager.beginNewChatSession(InfoDao.getInstance().getConnectionId()
        , IMChatManager.getInstance().getIsNewVisitor()
        , peerId
        , j2.toString()
        , getResponseListener( fromBeginSession: true));
}

```

2:ChatActivity 中的 beginScheduleSession() 方法 添加:

```

private void beginScheduleSession(String scheduleId, String processId, String currentNodeId, String entranceId) {
    if (IMChatManager.getInstance().getAppContext() == null) {
        return;
    }
    IMChatManager.userId = InfoDao.getInstance().getUserId();
    HttpManager.beginNewScheduleChatSession(InfoDao.getInstance().getConnectionId()
        , IMChatManager.getInstance().getIsNewVisitor()
        , scheduleId
        , processId
        , currentNodeId
        , entranceId
        , j2.toString()
        , getResponseListener( fromBeginSession: false));
}

```

注意：设置扩展信息一定要确保会话是一次全新的会话，即第一次进入会话设置才会有效。

#### 4、发送商品链接

##### 1: cardInfo

在初始化之前调用 `setCard(new CardInfo())`

查看示例代码

```
CardInfo ci = new CardInfo("图片 url", "第一行内容", "第二行内容", "第三行内容", "点击  
跳转链接");  
//设置 card  
helper.setCard(ci);  
//设置参数初始化  
helper.initSdkChat("com.m7.imkf.KEFU_NEW_MSG",  
    "请填写后台获取的 accessId",  
    "hhtest",  
    "userId");
```

如果商品信息中包含了特殊字符的话，特殊字符的地方要加上转义，为了兼容所有字段可能出现特殊字符，全部增加转码如下,为了前端可以正常展示，前端要相应的转码

```
try {  
    ci = new CardInfo(URLEncoder.encode(icon, "utf-8"), URLEncoder.encode(title, "utf-8"),  
        URLEncoder.encode(content, "utf-8"), URLEncoder.encode(rigth3, "utf-8"),  
        URLEncoder.encode(s, "utf-8"));  
} catch (Exception e) {  
    e.printStackTrace();  
}  
//设置 card  
helper.setCard(ci);
```

##### 2: NewCardInfo

在初始化之前调用 `handleNewCardInfo()`

查看示例代码

```
private void handleNewCardInfo() {  
    NewCardInfoTags tags1 = new NewCardInfoTags();  
    tags1.setLabel("按钮 1");  
    tags1.setUrl("https://www.baidu.com/?tn=64075107\_1\_dg");  
    tags1.setShowRange("all");//该字段不传在或者为 all 访客户端座席端都展示,seats 只座席  
    端展示, visitor 只访客户端展示
```

```
NewCardInfoTags tags2 = new NewCardInfoTags();
tags2.setLabel("按钮 2");
tags2.setUrl("https://www.sogou.com/");
tags2.setShowRange("all");//该字段不传在或者为 all 访客端座席端都展示,seats 只座席端展示, visitor 只访客端展示
```

```
ArrayList<NewCardInfoTags> tags = new ArrayList<>();
tags.add(tags1);
tags.add(tags2);
```

```
NewCardInfo newCardInfo = new NewCardInfo.Builder()
    .setTitle("我是标题")
    .setAttr_one(new NewCardInfoAttrs().setColor("#487903").setContent("x9"))
    .setAttr_two(new NewCardInfoAttrs().setColor("#845433").setContent("未发货"))
    .setOther_title_one("附件信息 1")
    .setOther_title_two("附件信息 2")
    .setOther_title_three("附件信息 3")
    .setSub_title("我是副标题")
    .setPrice("$999")
    .setImg("http://seopic.699pic.com/photo/40023/0579.jpg\_wh1200.jpg")
    .setTarget("https://kf.aiquismart.com/login")
    .setTags(tags)
    //setCardID 自定义一个卡片 id, 作用是当执行自动发送时, 可记录当前卡片是否
    发送过, 这个 id 的赋值规则客户可以完全自定义
    //如果使用自动发送那么建议就配置 cardID, 否则会重复发送
    .setCardID("")
    .setAutoCardSend("false");//是否自动发送 字符串 "false"否 "true"是
    .build();
```

```
IMChatManager.getInstance().setNewCardInfo(newCardInfo);
}
```

## 5、开启后主动会话（后台选择 sdk 模式）

- 关闭会话依旧保持长连接(demo 中为 [R.id.chat\\_tv\\_back](#) 的注销按钮,以及 onBackPressed 事件)


























```
IMChatManager.getInstance().onLineQuitSDK();
```

- 获取未读消息数

```
int unreadCount = IMChatManager.getInstance().getMsgUnReadCount();
```

注意：如果选用 sdk 的主动回话功能，因为安卓没有统一的离线推送机制，本解决方案只是在用户注销时，其实后台长连接进程还在，短时间坐席发送的消息可以收到，时间一长或者 app 重启，休眠都会造成后台进程断开；（用户可以采用 url 推送，借助 app 的推送机制进行推送消息（eg:华为离线推送、小米离线推送等））

## **6、本 sdk 支持中文和英文显示切换**

- ▶  .idea
- ▼  **app**
  - ▶  build
  - ▶  libs
  - ▼  src
    - ▶  androidTest
    - ▼  main
      - ▶  java
      - ▼  res
        - ▶  anim
        - ▶  drawable
        - ▶  drawable-v24
        - ▶  layout
        - ▶  mipmap-anydpi-v26
        - ▶  mipmap-hdpi
        - ▶  mipmap-mdpi
        - ▶  mipmap-xhdpi
        - ▶  mipmap-xxhdpi
        - ▶  mipmap-xxxhdpi
        - ▶  values
        - ▶  values-en
      - ▶  AndroidManifest.xml
    - ▶  test
  - ▶  .gitignore
  - ▶  app.iml

- 如需移植到项目中，请将 values-en 下的 string.xml 复制到项目中该文件下
- 默认 sdk 语言环境跟随手机系统变化
- 若需手动切换可在 sdk 初始化之前调用 demo MainActivity 提供的切换英文语言方法 initLanguage() 示例

也可以使用您 App 中原有的语言切换配置方法。

## 7、聊天关闭后获取消息未读数

```
/*
    获取未读消息数
    注意：调用此方法前也需要先配置服务
    @param accessId    接入 id（需后台配置获取）
    @param userName    用户名
    @param userId      用户 id
    */
IMChatManager.getInstance().getMsgUnReadCountFromService(
    accessId, userName, userId, new IMChatManager.HttpUnReadListen() {
        @Override
        public void getUnRead(int account) {
            Toast.makeText(MainActivity.this, getString(R.string.ykf_sdk_ykf_unreadmsg) +
" " + account, Toast.LENGTH_SHORT).show();
        }
    });
```

## 六、常见问题处理

### 1、点击录音按钮异常或者初始化报关于.so 的错误

so 库支持：

x0; 'armeabi'

'armeabi-v7a'

'arm64-v8a'

'x86'

'x86\_64'

配置方式：



在 ykf sdk 和 app 层的 defaultConfig 添加对应所需要的 abiFilters 即可

```
defaultConfig { DefaultConfig it ->
    minSdkVersion gradle.ext.minSdkVersion
    targetSdkVersion gradle.ext.targetSdkVersion
    versionCode gradle.ext.versionCode
    versionName gradle.ext.versionName
    multiDexEnabled true
    consumerProguardFiles 'moor-rules.pro'
    ndk { NdkOptions it ->
        // 设置支持的SO库架构
        abiFilters 'arm64-v8a', 'armeabi-v7a'
    }
}
```

离线资源：录音相关的 xa0; ([点击下载 so 库](#))

2、消息接收不显示，但在发送消息后同时显示之前接收的消息

可能原因：

检查 com.m7.imkfsdk.receiver.NewMsgReceiver 的文件是否和 demo 中的文件包名是否一致  
(因 jar 中广播指向的是该包名下的 NewMsgReceiver，android8.0 以后广播需携带包名)

3、客服 im 退出后，再次进入客服聊天界面异常、闪退

可能原因：在点击退出按钮是未添加退出 sdk 代码。

//退出聊天界面就注销了 SDK，若不需要注销则把该处代码去掉  
IMChatManager.getInstance().quitSDK();

4、sdk 集成后表情显示不出来

原因：1、assets 文件下表情对应关系 emojiKF 不存在

xa0; xa0; xa0; xa0; xa0; 2、未添加表情初始化代码

解决：将 demo 里的 emoji.kf 文件拷贝到项目中该目录下、添加初始化代码·

```
//初始化表情
new Thread(new Runnable() {
    @Override
    public void run() {
        com.m7.imkfsdk.utils.FaceConversionUtil.getInstance().
            getFileText(getApplicationContext());
    }
}).start();
```

5、商品发送链接客户端发送出去 pc 坐席端收不到

解决：将发送点击链接进行 utf-8 转码

```
CardInfo ci = null;
try {
    ci = new CardInfo("http://seopic.699pic.com/photo/40023/0579.jpg_wh1200.jpg",
        "我是一个标题当初读书",
        "我是 name 当初读书。",
        "价格 1000-9999",
        URLEncoder.encode(s, "utf-8"));
} catch (Exception e) {
    e.printStackTrace();
}
helper.setCard(ci);
```

6、sdk 初始化回调错误码说明

1001:初始化超时，检查网络环境或服务配置 1002:获取 tcp 或 ws 服务失败 1004:登录失败  
1005:消息发送失败未知错误

1006:初始化之前没有调用 RequestUrl.setRequestUrl()或 RequestUrl.setRequestUrl();

1007:进入会话没有携带技能组 id

1008:进入会话没有携带日程所需字段

x0;1009:进入会话没有携带会话类型

1010: isIniting 为 true，请确保 sdk 不是正在初始化状态

1011: 初始化之前没有配置 imageloader

1012: 初始化的 userName 为空 x0;

1013: 初始化的 userId 为空 x0;

#### 7、Android 是否提供支持 Androidx 的 sdk 包?

包含在下载 zip 文件中

#### 8、Android 集成 sdk 初始化后报错:

```
java.lang.NullPointerException: url == null
```

```
xa0; xa0; xa0; xa0;at okhttp3.Request$Builder.url(SourceFile:156)
```

检查在 helper.initSdkChat 之前是否调用了环境配置 RequestUrl.setRequestBasic();

#### 9、Android 混淆打包后进入会话页面报错:

如果代码设置了 release 或 debugxa0;打包混淆, 需要查看是否添加了 sdk 的混淆配置。

具体混淆配置请查看下载的 sdk 压缩包中提供的集成文档。

#### 10、Android 如何配置离线推送?

前往 pc 后台设置-app 设置-找到配置 App 名称。

点击修改配置您服务端的 Http 服务地址, 当访客离线后我们会推送消息到配置的服务地址。当您的后端收到这个消息后可以进行 App 分发推送。推送实现您可以保持项目原有推送逻辑。如果您的 App 之前没有推送功能那么需要新对接其他推送服务。

#### 12、Android 导入 sdk 后依赖包冲突

xa0;sdk 中使用了 Qiniu, gson, eventbus, glide 等依赖库, 如果集成导入后造成项目依赖冲突可以尝试以下两种方案:

xa0;1: 调整您项目的 moudle 依赖顺序, 项目使用 sdk 中所提供的依赖库版本。

xa0;2: 进行依赖排除, 使用 exclude 排除 sdk 中提供的依赖包。

#### 13、gradle 版本与 Android studio 不一致或版本过高

xa0;xa0;为保障良好的开发体验和更好的性能, 强烈建议保持 Android Studio 较新版本

xa0; sdk v3.6.0 以上最低支持 Plugin version 3.2.0+, Required Gradle version 4.6+ 以上版本

xa0; 如果您的项目还在使用 Plugin version 3.2.0 以下版本:

xa0; xa0;1: Gradle 3.2.0 发布时间为 2018.9 月, 可以满足大多数的新版三方类库以及 Android 各个版本的兼容, 强烈建议您升级 Android Studio 或使用新版的 gradle 编译工具

xa0; xa0;2: 可以操作依赖排除本 sdk 中的 Gson 库, 重新集成一个较低版本的 Gson 库, 因为 sdk 使用 Gson 推荐版本 2.8.6 版本不支持 3.2.0 以下编译工具。

#### 14、video-call-sdk 依赖下载报错

音视频 SDK 遇到 video-call-sdk 依赖无法下载或下载困难, 请查看网络是否可以正常访问 github, 可以上网获取 github host 修改方案调整电脑的网络配置, 或将 maven 仓库更换为测试可用的镜像加速仓库。

#### 15、AndroidX 版本项目运行报错 Execution failed for task ':app:checkDebugDuplicateClasses'.

```
• What went wrong:
Execution failed for task ':app:checkDebugDuplicateClasses'.
> A failure occurred while executing com.android.build.gradle.internal.tasks.CheckDuplicatesRunnable
  > Duplicate class android.support.v4.app.INotificationSideChannel found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.app.INotificationSideChannel$Stub found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.app.INotificationSideChannel$Stub$Proxy found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.os.IResultReceiver found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.os.IResultReceiver$Stub found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.os.IResultReceiver$Stub$Proxy found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.os.ResultReceiver found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.os.ResultReceiver$1 found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.os.ResultReceiver$MyResultReceiver found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)
    Duplicate class android.support.v4.os.ResultReceiver$MyRunnable found in modules core-1.7.8-runtime (androidx.core:core:1.7.8) and support-compat-27.1.1-runtime (com.android.support:support-compat:27.1.1)

  Go to the documentation to learn how to <a href="d.android.com/r/tools/classpath-sync-errors">fix dependency resolution errors</a>.
```

遇到图中类似报错, 请在 gradle.properties 中添加以下代码

android.enableJetifier=true

#### 16、添加 sdk 后三方异常监控 sdk 无法收集信息

sdk 内部使用 Thread.UncaughtExceptionHandlerx0;类用于获取部分 sdk 在使用中的时候发生的异常问题。

如果在与其他三方 crash sdk 共同使用遇到一些特殊问题, 比如: 三方不上报、三方报错、其他异常等。

可以关闭 sdk 的错误统计。使用以下代码:

```

super.onCreate();

/**
 * 1 (重要):
 * 请检查在 Application onCreate() 中的初始化
 * 初始化YKFUtils : YKFUtils.init(this)
 *
 * 注意:如果使用三方crash监控等插件,请在初始化三方crash监控之前init本sdk,像下面demo这样的顺序.
 */
YKFUtils.AddCrashHandler=false;
YKFUtils.init(application: this);

//bugly(demo 演示用,可移除)

```

17、项目报错 Attempt to invoke virtual method 'android.content.res.XmlResourceParser [android.content.pm.ProviderInfo.loadXmlMetaData\(android.content.pm.PackageManager, java.lang.String\)](#)' on a null object reference

FileProvider 配置的 authorities 与代码中的不同导致的

- 1: 请参考 demo FileProvider authorities 配置
- 2: 在实际项目中查看当前的 FileProvider authorities 是否一致