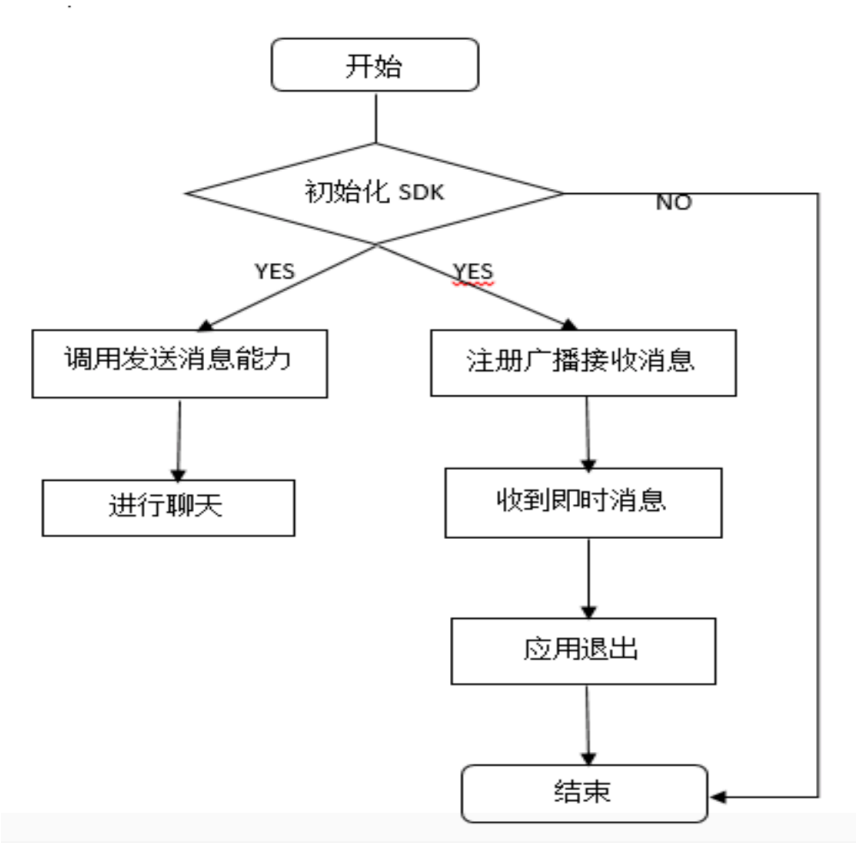
客服IMSDK开发指南（Android）

一、sdk工作流程



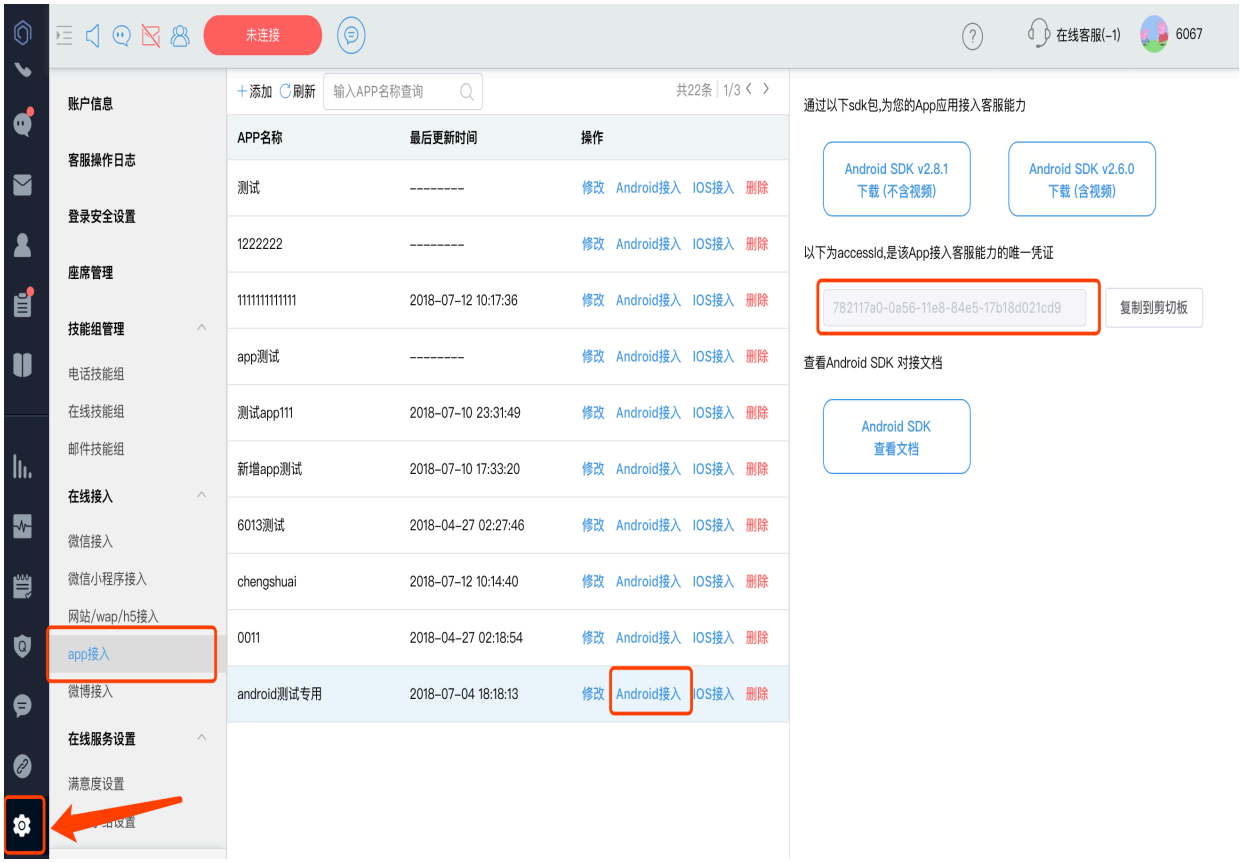
二、sdk下载文件说明

1、sdk文件包含内容：

- 所需jar包
- 录音所需so库（demo 中只含有部分主要平台）
- 对接说明文档
- 运行事例demo(“KEFUDEMO”文件夹)

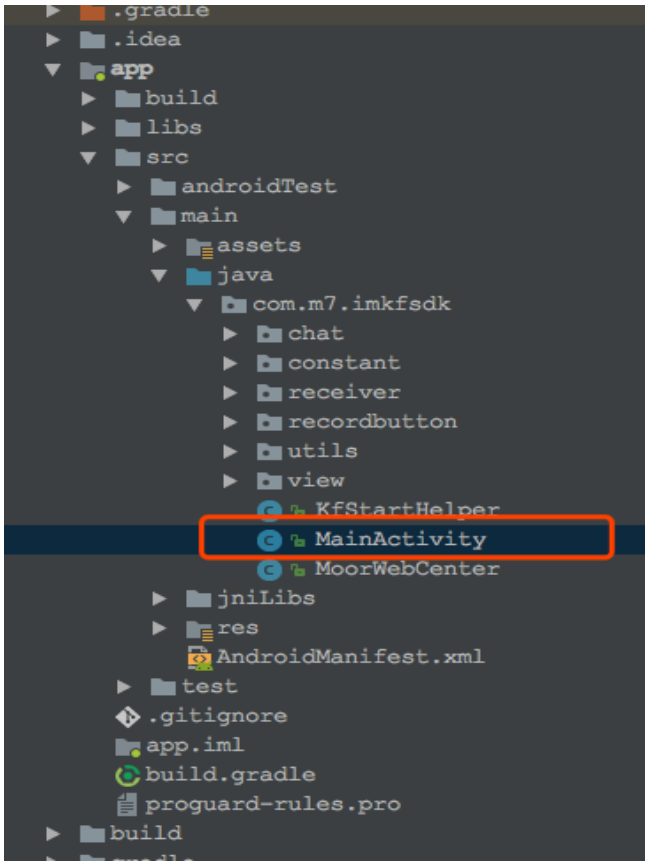
2、accessId获取方法

登录呼叫中心网站，点击左侧最下面的设置，选择右侧面板中的渠道接入配置，选择移动客服配置，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。



3、运行KEFUDEMO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("com.m7.imkf.KEFU_NEW_MSG",
                   "请填写后台获取的accessId",
                    "hhtest",
                    "userId");
```

(填写获取的accessId后可直接运行demo,查看效果)

三、开始集成

1、导入资源

- 将IMKFSDK-x.x.x.jar导入到项目中的libs目录中。
- 若使用发送录音功能，还需复制jniLibs目录中的so包，demo中包含主流格式，复制自己所需格式的即可（如需所有格式点击下载so库）
- 修改 AndroidManifest.xml 文件

  (在AndroidManifest.xml文件中添加相关权限以及相关代码，具体查看demo)

```
  <service   android:name="com.moor.imkf.tcpservice.service.IMService">
  </service>
  <receiver  android:name="com.moor.imkf.receiver.NetWorkReceiver">
            <intent-filter>
                <action android:name="android.net.conn.CONNECTIVITY_CHANGE"
/>
            </intent-filter>
  </receiver>
```

2、复制初始化代码

将demo中的资源文件及相关代码文件复制到自己的项目中

重要：需要修改的内容

1、MianActivity.java中初始化SDK接口中填写action,accessId等相关参数AndroidManifest.xml中的消息receiver中，action的name中填写初始化SDK时所填的action，必须保证该action的值一致，否则接收不到消息。 若要修改其中的相关布局资源

2、头像： head_default_local.png为自己头像，head_default_robot.png为客服头像，若需替换直接替换这2张图片即可。

3、使用demo时须初始化聊天界面表情功能

```
new Thread(new Runnable() {
            @Override
            public void run() {
com.m7.imkfsdk.utils.FaceConversionUtil.getInstace().
            getFileText(getApplicationContext());
            }
 }).start();
```

3、接口说明

初始化SDK

```
IMChatManager.getInstance().init(getApplicationContext(),
             "receiverAction",
             "accessId",
             "username ",
             "userId");
```

其中参数说明：
Context context, 应用上下文.
String receiverAction 注册接收消息广播的action，填写自己定义的，该值也得填写到.
AndroidManifest.xml中的NewMsgReceiver中的action中。
String accessId, 接入号,必填项.

String username, 用户名，用来在后台显示.
String userId, 用户id，用来标识用户.

SDK初始化的接口监听

```
MChatManager.getInstance().setOnInitListener(new InitListener() {
        @Override
        public void oninitSuccess() {
            LogUtil.d("MobileApplication", "sdk初始化成功");
        }
        @Override
        public void onInitFailed() {
            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

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().
getConnectionId(), new GetGlobleConfigListen() {
            @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。

从服务器获取消息未读数

用户离开后，坐席向客户发送消息的未读数

```
IMChatManager.getInstance().getMsgUnReadCountFromService(new
                    IMChatManager.HttpUnReadListen() {
                         @Override
                         public void getUnRead(int acount) {

                         }
                     });
```

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.sendEmptyMessage(0x111);
            }else if(IMChatManager.ONLINE_ACTION.equals(action)) {
                //当前客服在线
                handler.sendEmptyMessage(0x222);
            } else if(IMChatManager.OFFLINE_ACTION.equals(action)) {
                //当前客服不在线
```

```
                handler.sendEmptyMessage(0x333);
            }
        }
    }
```

具体请参看demo；

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：

```
IMChatManager.getInstance().convertManual(new 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:

```
IMChatManager.getInstance().submitOfflineMessage(peerId, content, phone,
email, new OnSubmitOfflineMessageListener() {
                            @Override
                            public void onSuccess() {
                                dismiss();
                                Toast.makeText(getActivity(), "提交留言成功",
Toast.LENGTH_SHORT).show();
                            }
                            @Override
                            public void onFailed() {
                                dismiss();
                                Toast.makeText(getActivity(), "提交留言失败",
Toast.LENGTH_SHORT).show();
                            }
 });
```

提交离线留言中参数peerId为技能组的id, content为留言内容,phone是电话,email是邮箱。

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：目前有4种，分别为文本消息，录音消息，图片消息，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

拼装一条消息

- 文本消息：

使用如下代码

```
FromToMessage fromToMessage = IMMessage.createTxtMessage(txt);
```

参数说明： String txt, 消息文本内容.

- 录音消息:

使用如下代码.

```
FromToMessage fromToMessage = IMMessage.createAudioMessage(mTime,
filePath);
```

参数说明： float mTime, 录音时长 String filePath, 录音在本地的路径

- 图片消息:

使用如下代码

```
FromToMessage fromToMessage =
IMMessage.createImageMessage(picFileFullName);
```

参数说明： String picFileFullName, 图片在本地的路径

- 文件消息：

```
FromToMessage fromToMessage = IMMessage.createFileMessage(path, fileName,
fileSizeStr);
```

参数说明： String path, 文件在本地的路径 String fileName, 文件名字 String fileSizeStr, 文件大小 拼装好的消息在发送时用到。

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

录制MP3格式语音

由于跨平台的需要，发送的录音文件格式需为mp3格式，因此提供了录制mp3格式的方法，代码如下：

```
MP3Recorder mp3Recorder = new MP3Recorder(file);//初始化录音器
mp3Recorder.start();//开始录制
mp3Recorder.stop();//结束录制
```

其中的参数为：File file录音保存文件,具体使用可参看提供的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，首先需要一个全局注册的广播，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

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：

```
List<FromToMessage>fromToMessages=IMChatManager.getInstance().getMessages(1
);
```

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。

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().updateMessageToDB(message);
```

参数为FromToMessage message,修改数据后的消息。

判断数据库中数据是否都取出

界面显示时需要判断本地数据库中的数据是否已经被全部取出，代码如下：

```
IMChatManager.getInstance().isReachEndMessage(size);
```

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。

注销

若需注销当前用户，则可以调用该接口，代码如下：

```
IMChatManager.getInstance().quit();
```

设置IP地址(私有云用户)

若需设置自己服务器的地址（在初始化之前调用以下方法），代码如下：

```
//设置tcp地址
IMChatManager.getInstance().setTcpIpAndPort("11.111.1.11", 8006);
//设置http地址
IMChatManager.getInstance().setHttpIp("http://11.111.1.11:4999/sdkChat");
```

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();
```

4、混淆

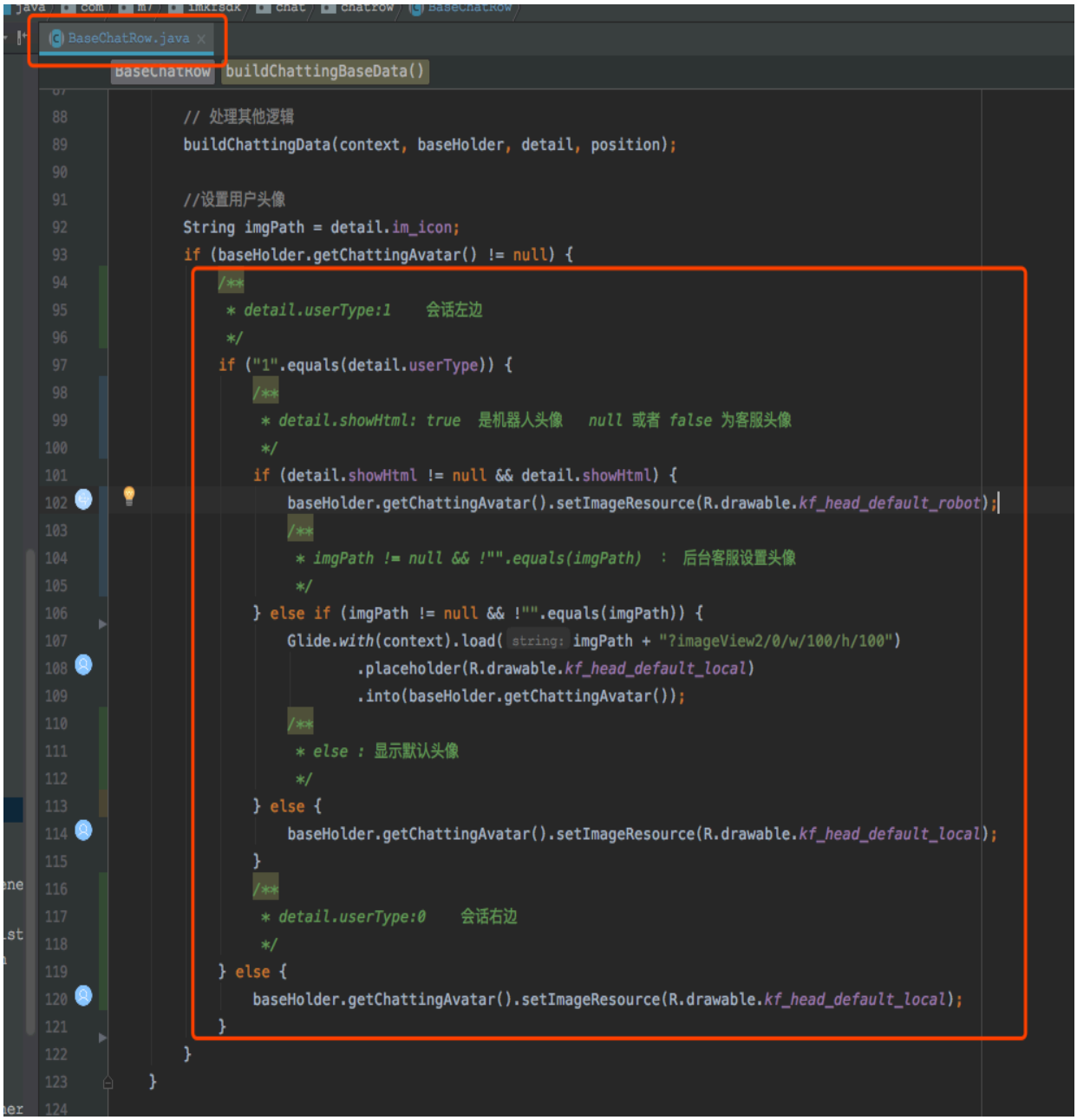
混淆时加上如下代码：

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

shrinkResources设置为false，不然表情显示不正确

四、定制功能

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

调用beginSession的重载方法，可传入json字符串，示例如下：

```
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

调用beginSession的重载方法，可传入json字符串，示例如下

```
        JSONObject j = new JSONObject();
        JSONObject j2 = new JSONObject();
        try{
            j.put("name","测试扩展字段");
            j2.put("customField",URLEncoder.encode(j.toString()));
        }catch ( e){
        }
    IMChatManager.getInstance().beginSession(peerId ,j2.toString(),new
OnSessionBeginListener() {});
```

4、发送商品链接

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");
```

5、开启后主动会话（后台选择sdk模式）

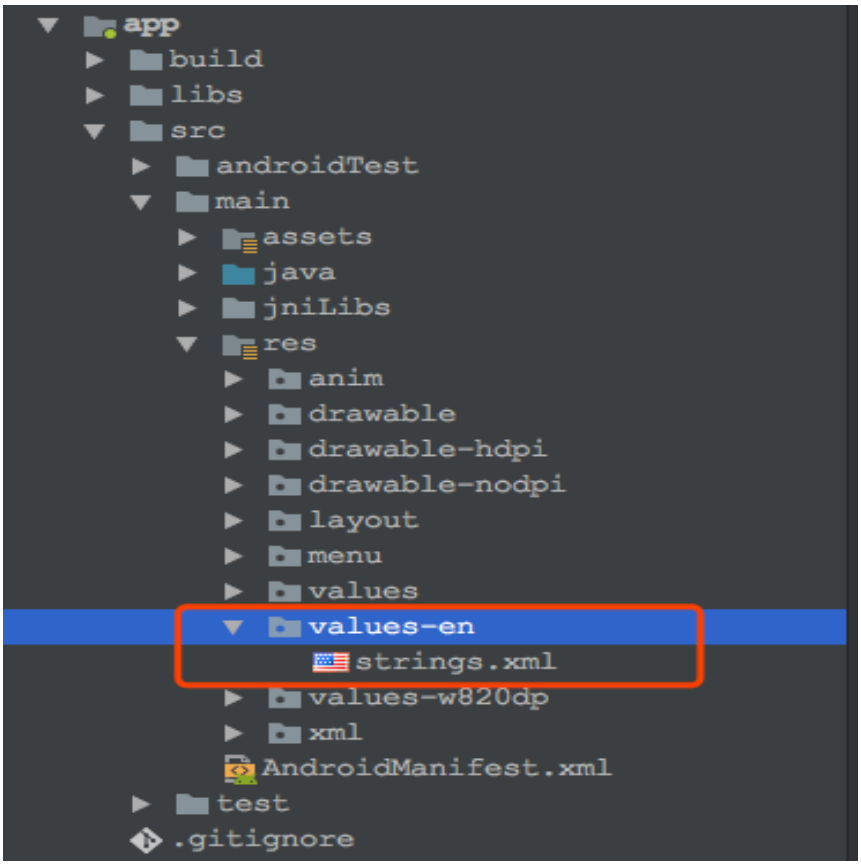
- 关闭会话依旧保持长连接(demo中为R.id.chat_tv_back的注销按钮)

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

- 获取未读消息数

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

6、本sdk支持国际英文版



- 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
- 若需切换可根据需求在界面加载之前（全局可在项目application类中）执行以下代码：

```
//加载默认 中文
initLocaleLanguage（“”）；
//记载英文
initLocaleLanguage（“en”）；

private void initLocaleLanguage(String language) {
        Resources resources = getApplicationContext().getResources();
        Configuration configuration = resources.getConfiguration();
        configuration.locale = new Locale(language);
        resources.updateConfiguration(configuration,
resources.getDisplayMetrics());//更新配置
    }
```

五、常见问题处理

1、导入客服sdk后发现报错（项目报错，sdk demo 运行正常）缺少 .so 库

可能原因：一般是因为每个平台下的so库数量、种类不统一，只要保证每个平台下的so库数量一致，即可决绝问题;
解决：根据项目原有目录添加对应so库 （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：

1、接受消息的service注册时的action和监听时的action不一致 （初始化参数1和清单文件的receiver的action是否一致）

2：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().quit();
IMChatManager.isKFSDK = false;
```

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

2、未添加表情初始化代码

解决：将demo里的emojikf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.getInstace().
            getFileText(getApplicationContext());
            }
 }).start();
```

六、版本说明

2.7.0更新日志

- 取消demo里面关于application 的部分（demo中MobileApplication文件）
- 之前所有的 MobileApplication.isSDK 的判断全改为IMChatManager.isKFSDK
- 将之前MobileApplication类中的 表情的初始化 移到 MainActivty 类
- 优化部分代码
- 修复发现bug
- 删除多余文件，减少体积

2.7.1更新日志

- 支持国际英文版
- 优化部分代码
- 修复已发现异常

2.8.0更新日志

- 添加主动会话功能
- 优化部分代码
- 修复一发现bug

2.8.1更新日志

- 调整代码入口代码
- 调整录音、照相权限申请位置
- 添加商品链接功能
- 修复android8.0以后广播和通知接受异常