V5 智能客服 Android 客户端 SDK 接口

（Ver0.7）

目录·

1 术语

应用账号： V5KF 网站后台 AppSDK 应用配置中的应用账号。

站点编号：V5KF 网站的账号对应的站点编号（可以在V5KF 官网后台查看或向客服获取）。

Demo 工程： 使用智能客服系统 SDK 客户端开发的接口使用示例工程。

会话界面： 针对使用本 SDK 的 Android APP 而言，表示进行对话的一个 Activity。

device_token： 推送平台用于标识设备的唯一 ID，长度为 64 字节以内的字符串。

用户 ID(uid)： 标识 APP 所登录的用户的唯一 ID，长度为 64 字节以内的字符串。

2 功能说明

V5 智能客服系统客户端可集成到 web、APP 等第三方平台提供客户在线咨询服务，实时接收客户的反馈。支持发送文本、位置、图片以及表情等消息，并可显示图文、打开链接。

本文档介绍 V5 智能客服系统客户端 SDK 的 Android 版本的集成和使用。本 SDK 兼容 Android API 9 以上，并为开发者提供源码和 Demo 工程，可以参照 Demo，使用 SDK 提供的 UI 快速集成到你的项目中；对 UI 有较高定制需求的开发者可根据 SDK 接口进行开发，自行开发界面。

3 业务流程

3.1 业务结构

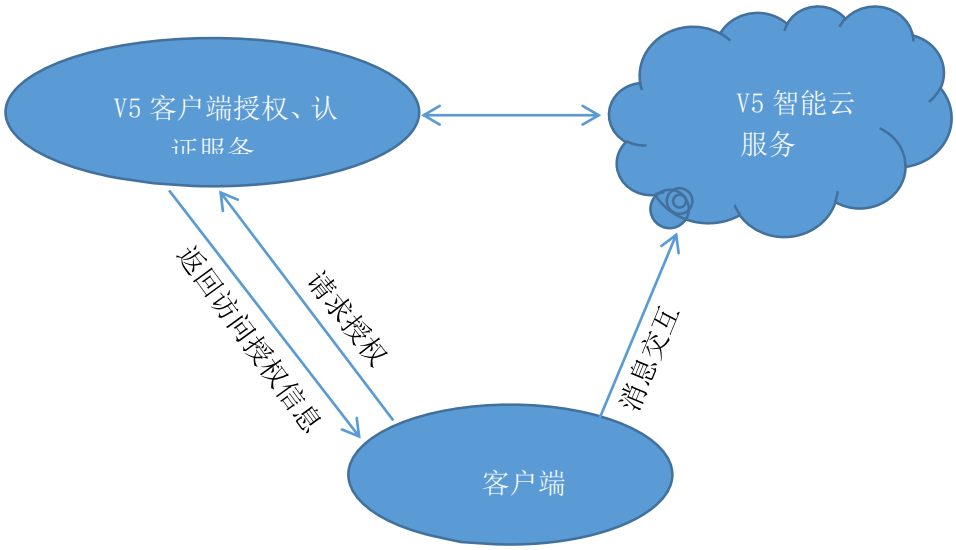

图 3-1 应用结构

- V5 智能云服务

V5 智能云服务，是连接座席和访客的桥梁。并通过云智能机器人，提供替代、协助座席进行优质客服的服务。

- V5 客户端授权、认证服务

分发访客接入 V5 智能云服务的凭据服务器。

3.2 交互流程

- 客户端首先向 V5 客户端授权、认证服务发送认证信息（HTTP POST 请求方式），以获取连接 V5 智能云服务的授权信息；
- 用认证成功返回的授权信息向 V5 智能云服务建立会话连接；
- 开启会话，进行即时消息对话。

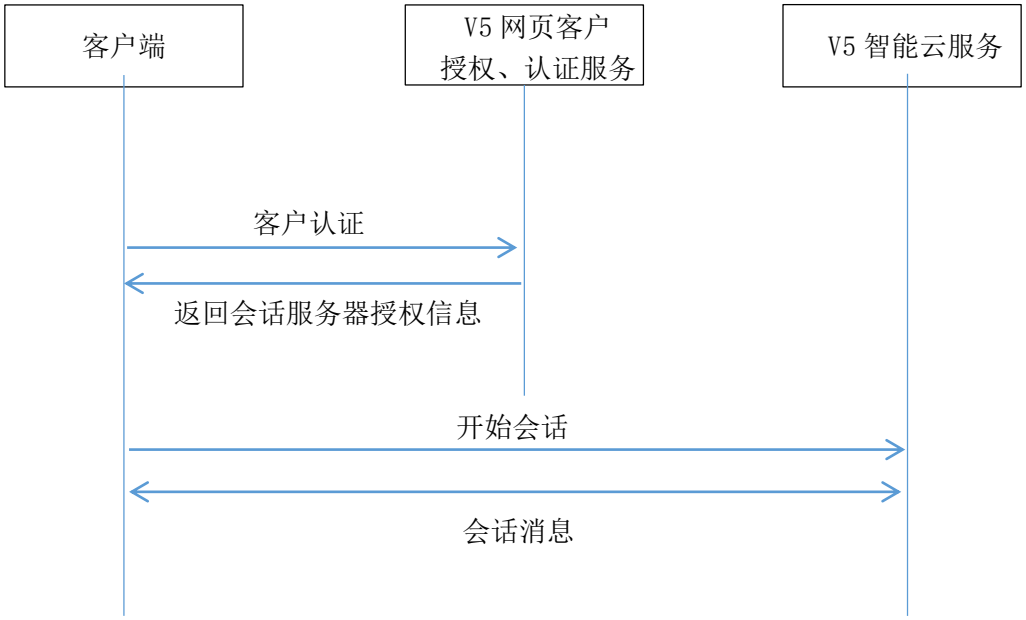

图 3-2 交互流程

3.3 SDK 工作流程

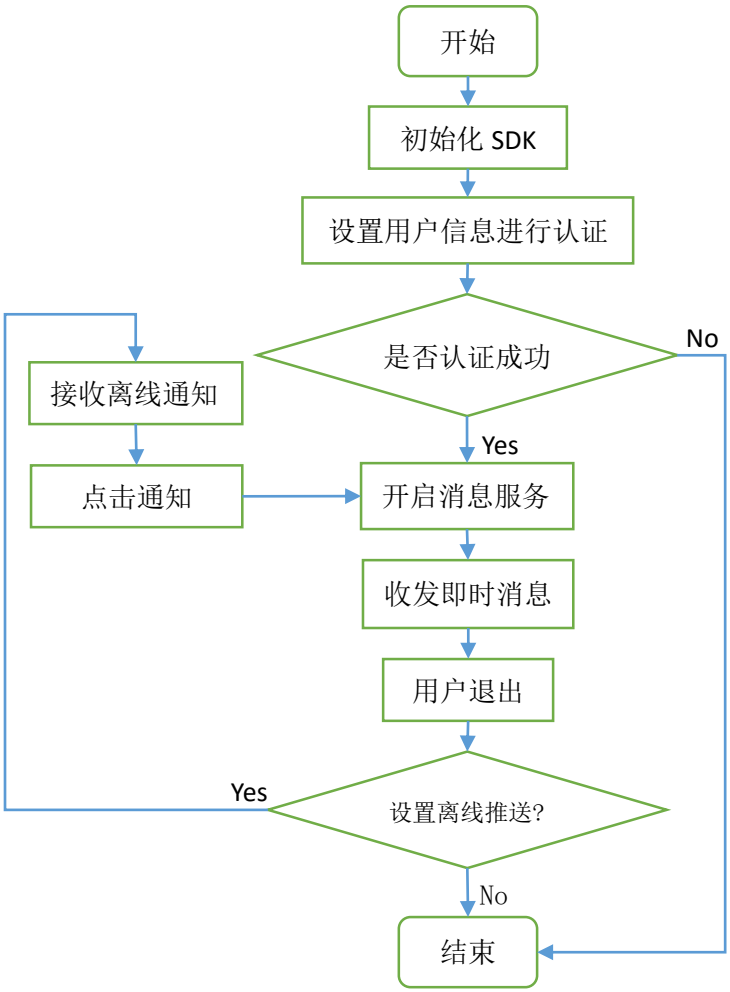

图 3-3 SDK 工作流程

4 前期准备

4.1 开发环境准备

1. V5KF 客服系统账号

   没有 V5KF 账号需要前往官网注册账号。

2. 获得应用账号、站点编号

   应用账号、站点编号作为 SDK 连接服务端的身份凭证，可到后台 App SDK 的应用配置界面获取。

3. 填写对应平台的推送服务器地址

   为了使您的 APP 在集成本 SDK 后具有离线消息推送，建议填写您的推送服务器地

址，同时也支持第三方推送平台，需要按照本文档规定填写您的 device_token 和绑定的用户 ID。

4. 下载 SDK

您可以到 V5KF 官网下载智能客服 SDK，包含了开发包和带 UI 界面的 Demo 示例工程。

5. 环境要求

在您集成智能客服 SDK 前环境要求如下：

- Android SDK Build-tools 请升级到 21 及以上版本。
- JAVA 编译版本 JDK 1.7 及以上版本。
- 编译 Demo 需要 Android Support V7 21 及以上版本(需导入支持包 android-support-v7-appcompat)。

Android SDK 最低支持 Android API 9: Android 2.3(Gingerbread)。

4.2 配置 AndroidManifest

可以参考 Demo 工程的 AndroidManifest.xml 文件来配置您的 AndroidManifest，无论是使用 SDK 的接口开发还是直接使用 Demo 工程的 UI 快速集成都需要对您的项目的 AndroidManifest.xml 文件进行下述配置，具体配置项目如下：

1. 配置站点信息

```
<meta-data android:value="您的站点编号" android:name="V5_SITE" />

<meta-data android:value="您的应用账号" android:name="V5_ACCOUNT" />
```

2. 添加必需的权限

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

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

<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

若使用腾讯地图模块，还需添加：

```
<!-- 通过 GPS 得到精确位置 -->

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

<!-- 通过网络得到粗略位置 -->

<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

```
<!-- 访问 WiFi 状态. 需要 WiFi 信息用于网络定位 -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
/>
<!-- 修改 WiFi 状态. 发起 WiFi 扫描, 需要 WiFi 信息用于网络定位 -->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"
/>
<!-- 访问网络的变化, 需要某些信息用于网络定位 -->
<uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />
<!-- 访问手机当前状态, 需要某些信息用于网络定位 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE"
/>
```

3. 配置使用自定义的 Application

在 Application 的 onCreate 中需要进行 SDK 的初始化，故需要自定义自己的 Application 类，并在 AndroidManifest.xml 中进行下面配置：

```
<application
        android:allowBackup="true"
        android:name="com.your.package.您的自定义 Application 类"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <!—其他内容 -->
</application>
```

4. 添加必需的服务和 Activity

```
<service
     android:name="com.v5kf.client.lib.V5ClientService" >
</service>

<!—使用不带 UI 的 SDK 接口进行开发可以省略下面的配置 -->
<activity
     android:name="com.v5kf.client.ui.LocationMapActivity" >
</activity>
<activity
```

```
     android:name="com.v5kf.client.ui.WebViewActivity" >
</activity>
<activity
     android:name="com.v5kf.client.ui.ShowImageActivity" >
</activity>
```

5. 会话 Activity 及 intent-filter 配置

```
<activity
    android:name="com.v5kf.client.ui.ClientChatActivity"
    android:label="@string/ v5_chat_title"
    android:launchMode="singleTask"
    android:windowSoftInputMode="adjustResize" >
        <!-- 配置消息通知点击后跳转的intent-filter -->
        <intent-filter>
        <action
        android:name="com.v5kf.android.intent.notification105723" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

注意上面的配置中 action 的值为*"com.v5kf.android.intent.notification" + 您的**站点编号***，用于响应通知栏消息点击以打开会话界面。

4.3 了解离线消息推送

客户离线后消息将推送到您指定的推送服务器或者第三方平台，需要在 V5 官网后台做对应配置，选择对应推送平台并配置，否则离线后接收不到消息：

1. 您自己的推送服务器：

应用配置须填写：推送服务器接口地址，V5 系统将 POST 离线消息到您的服务器接口。

APP 代码中需设置：V5ClientConfig.getInstance().setDeviceToken("识别您的推送终端的唯一 ID")。

2. 第三方推送平台：

1) 信鸽推送：

应用配置须填写：ACCESS ID 和 SECRET KEY

APP 代码中需设置：V5ClientConfig.getInstance().setDeviceToken("信鸽 SDK 中获取到的 token")。

2) 百度云推送：

应用配置须填写：API KEY 和 SECRET KEY

APP 代码中需设置：V5ClientConfig.getInstance().setDeviceToken("百度云 SDK 中获取到的 channel_id")。

3) 其他平台待补充。

注：从第三方推送平台接收到消息会附带有自定义参数”v5_action”:”new_message”（键值对），在对应的推送 SDK 接口中获取，以识别此消息来自 V5 智能客服。

5 使用 SDK 提供的 UI 快速集成

5.1 导入文件

- 将 SDK 压缩包中的 res 文件夹复制到你项目的对应 res 文件夹下；
- 根据是否使用带腾讯地图模块的开发包，进行下面二选一操作：

1. 使用腾讯地图模块：

1) 将 SDK 压缩包内的 V5KF_1.x.x_rxxxx.jar 复制到你的项目的 libs 文件夹下；

2) 将 SDK 压缩包中的 libs 文件夹下，腾讯地图的 SDK 包导入你项目的对应的 libs 文件夹下；

3) 在复制过的 res 目录下的 values 文件夹内的 v5_arrays.xml 打开，确保包含如下内容：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="v5_chat_func">
        <item>常见问题</item>
        <item>相关问题</item>
         <item>图片</item>
        <item>拍照</item>
        <item>人工客服</item>
        <item>位置</item>
    </string-array>

    <string-array name="v5_chat_func_icon">
        <item>v5_icon_ques</item>
        <item>v5_icon_relative_ques</item>
        <item>v5_icon_photo</item>
        <item>v5_icon_camera</item>
```

```
        <item>v5_icon_worker</item>
        <item>v5_icon_location</item>
    </string-array>
</resources>
```

2. 不使用腾讯地图模块：

   1) 将 SDK 压缩包内的 V5KF_noMap_1.x.x_rxxxx.jar 复制到你的项目的 libs 文件夹下；

   2) 在复制过的 res 目录下的 values 文件夹内的 v5_arrays.xml 打开，删除“<item>位置</item>”和“<item>v5_icon_location</item>”，确保内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="v5_chat_func">
        <item>常见问题</item>
        <item>相关问题</item>
        <item>图片</item>
        <item>拍照</item>
        <item>人工客服</item>
    </string-array>

<string-array name="v5_chat_func_icon">
        <item>v5_icon_ques</item>
        <item>v5_icon_relative_ques</item>
        <item>v5_icon_photo</item>
        <item>v5_icon_camera</item>
        <item>v5_icon_worker</item>
    </string-array>

</resources>
```

注：上述文件名称中的“x”表示 0~9 中某一数字，表示版本代号，下同。

5.2 初始化 SDK

初始化需要在您自定义的 Application 中执行，示例如下：

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();
        V5ClientAgent.init(this, new V5InitCallback() {

            @Override
            public void onSuccess(String response) {
```

```
                // TODO Auto-generated method stub
                Log.i("MyApplication", "init success: " + response);
            }

            @Override
            public void onFailure(String response) {
                // TODO Auto-generated method stub
                Log.e("MyApplication", "init failed: " + response);
            }
        });
    }
}
```

5.3 消息推送设置

推送参数设置：

```
V5ClientConfig config = V5ClientConfig.getInstance(Context context);

config.setDeviceToken("device_token 字符串"); // 【必须】,否则离线无法接收通知，离线消息通知发送到在 V5KF 后台配置的对应推送服务器地址，或者第三方推送平台
```

推送消息接受会收到后，在通知栏点击时需要打开客服会话界面。若您使用第三方推送平台将会返回一组自定义参数 ”v5_action” : ”new_message”，以此区分是否是来自 V5 智能客服系统的消息；若您使用自己的推送服务器，您可以在接收到消息后自行标记此消息并推送到客户端 APP（根据 device_token 识别接收客户端），进行启动客服会话界面处理，启动会话界面参考 5.5。

5.4 用户信息和参数设置

使用 SDK 提供的 UI 集成，需要在**启动会话界面**之前进行用户信息和参数配置。配置项如下：

```
// V5客服系统客户端配置
V5ClientConfig config = V5ClientConfig.getInstance(Context context);
config.setShowLog(true); // 是否打印日志，默认为true
config.setLogLevel(V5ClientConfig.LOG_LV_DEBUG); // 日志级别默认为全部显示
config.setUid("用户ID字符串"); // 【必须】,设置用户ID，区分APP登录的不同账号
config.setNickname("用户昵称"); // 设置用户昵称
config.setGender(1); // 设置用户性别
config.setAvatar("用户头像URL"); // 设置用户头像URL
config.setDefaultServiceByWorker(false); // 是否默认转人工客服
```

此外，对 SDK 中界面上面的内容操作也可以设置自定义的处理方式，如设置链接点击事件处理以及地图位置消息点击事件，接口如下。

设置 URL 链接点击监听：

```
V5ClientAgent.getInstance().setURLClickListener(OnURLClickListener
listener);
```

设置地图位置图片点击监听：

```
V5ClientAgent.getInstance().setLocationMapClickListener(OnLocationMap
ClickListener listener);
```

5.5 启动会话界面

通过简单地添加一个在线咨询按钮即可使用智能客服客户端功能，在按钮点击事件处理中加入：

```
// 开启对话界面
V5ClientAgent.getInstance().startV5ChatActivity(getApplicationContext
());
```

6 使用 SDK 接口开发

6.1 导入 libs 库

将下载的 SDK 压缩包内 CoreLib 目录下的 V5KF_core_1.x.x_rxxxx.jar 文件复制到您的项目 libs 目录下。

6.2 代码快速集成

在会话界面需要添加 SDK lib 中的代码，将消息服务集成到您的项目中。具体接口调用和代码添加如下。

6.2.1 初始化 SDK

初始化需要在您自定义的 Application 中执行，示例如下：

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();
        V5ClientAgent.init(this, new V5InitCallback() {

            @Override
            public void onSuccess(String response) {
                // TODO Auto-generated method stub
```

```
                Log.i("MyApplication", "init success: " + response);
            }

            @Override
            public void onFailure(String response) {
                // TODO Auto-generated method stub
                Log.e("MyApplication", "init failed: " + response);
            }
        });
    }
}
```

6.2.2 开启消息服务

在会话界面 Activity 的 onCreate 中调用：

```
V5ClientAgent.getInstance().start(Context context,
            V5MessageListener listener); // 开启消息服务
```

其中 V5MessageListener 是消息回调监听器：

```
public interface V5MessageListener { // 由会话 Activity 实现此接口
    public void onConnect(); // 会话连接建立成功，此回调成功后才可以调用其他消息接口
    public void onMessage(String json); // 接收消息为 json 字符串(兼容后期接口类型扩展)
    public void onMessage(V5Message message); // 接收消息对象
    public void onError(V5KFException error); // 返回异常信息
}
```

初始化内容包括：

1. 设置消息回调监听器；
2. 向 V5 认证服务进行客户端认证，根据 AndroidManifest 配置的站点信息和客户端生成的客户 ID 向认证服务器认证（这之前可进行客户端用户信息设置，参见 6.2.6），获取会话参数；
3. 认证成功并返回参数后开启 V5ClientService 消息服务。

6.2.3 消息接口调用

发送消息调用：

```
V5ClientAgent.getInstance().sendMessage (V5Message message,
        MessageSendCallback callback);
```

发送消息的第一个参数是 V5Message 消息对象，支持发送文本消息、位置消息、图片消息、控制消息：

```
// 新建文本消息
V5TextMessage textMsg =
V5MessageManager.getInstance().obtainTextMessage("消息内容");
// 新建位置消息
V5LocationMessage locationMsg = V5MessageManager.getInstance().
        obtainLocationMessage(
                double latitude,        // 纬度
                double longitude,       // 经度
                double accuracy,        // 精度,可为0
                String address);        // 地址描述，可为null
// 新建图片消息——发送本地图片
V5ImageMessage imageMsg =
V5MessageManager.getInstance().obtainImageMessage(
               String filePath);       // 本地图片路径
// 新建图片消息——发送网络图片
V5ImageMessage imageMsg =
V5MessageManager.getInstance().obtainImageMessage(
               String pic_url,         // 图片URL
               String media_id);       // 媒体ID，可为null
// 新建控制消息
V5ControlMessage controlMsg =
V5MessageManager.getInstance().obtainControlMessage(
                int code,              // 代码
                int argc,              // 参数数量，可为0，即参数可为空
                String argv);          // 参数值（字符串），可为null
// 目前开放的控制消息为转人工客服消息：code = 1，参数为空，示例如下：
V5Message msg = V5MessageManager.getInstance().obtainControlMessage(1,
0, null);
```

发送消息的第二个参数 MessageSendCallback 是消息发送结果回调：

```
  public interface MessageSendCallback {
    public void onSuccess(V5Message message); // 成功
    public void onFailure(V5Message message, int statusCode, String
desc); // 失败
  }
```

此外，当客户不在会话界面时，会发出消息通知，您可以通过注册 action 为"com.v5kf.android.intent.action_message"的广播接收器，来接收消息，通过下面方法从 onReceive 传递过来的 Intent 中获得消息对象：

```
V5Message message =
 (V5Message) intent.getSerializableExtra("v5_message");
```

在退出会话界面或者 APP 进入后台的时候可以调用：

```
V5ClientAgent.getInstance().onDestroy(); // 关闭消息服务
```

关闭消息服务表示用户下线，即无法继续接收消息，当用户再次进入会话界面时才能继续消息请求和接收，用户下线后消息将会缓存。若您设置了离线消息的推送服务器地址，您的服务器将会收到用户下线后的离线消息，可选择自行推送到您的 APP，或者保存到您的服务端，此外可选择对接到第三方推送平台，目前已支持腾讯信鸽推送和百度云推送。

此外，转人工客服可调用：

```
V5ClientAgent.getInstance().switchToArtificialService(MessageSendCallback callback);
```

6.2.4 生命周期处理

添加到会话界面 Activity 中 onStart、onStop 和 onDestroy 中处理的代码，用于判断接收到消息是否应该发出通知栏通知，当用户处于会话界面时无需发送通知栏通知。

```
@Override
protected void onStart() {
  super.onStart();
  V5ClientAgent.getInstance().onStart(); // 通知消息服务 onStart
}

@Override
protected void onStop() {
  super.onStop();
  V5ClientAgent.getInstance().onStop(); // 通知消息服务 onStop
}

@Override
protected void onDestroy() {
     super.onDestroy();
     V5ClientAgent.getInstance().onDestroy();// 通知消息服务onDestroy
}
```

6.2.5 通知设置

当后台推送设置为不推送时，在对话界面 onDestroy 前可以接收消息通知，通知在通知

栏显示，点击可进入对话界面。

```
ClientNotificationBuilder.setNotificationTitle(String title); // 默认为 V5 后台设置的推送标题

ClientNotificationBuilder.setSmallIcon(int mSmallIcon); // 通知右下角小图标(默认为 App 图标)

ClientNotificationBuilder.setLargeIcon(int mLargeIcon); // 通知左侧大图标(默认为 App 图标)

ClientNotificationBuilder.setTicker(String mTicker); // 通知滚动提示文字(默认为通知内容)

ClientNotificationBuilder.setIntent(Intent mIntent); // 设置点击本地通知跳转 Intent(默认为跳转到 action 为"com.v5kf.android.intent.notification + 站点编号"的 Activity, 携带包含键为”v5_message”值为 V5Message 对象的 Intent)
```

6.2.6 用户信息设置

V5ClientConfig 的配置需要在执行 SDK 的 start 方法初始化之前设置，非必须，但设置易识别的用户信息有助于客服识别客户，具体代码如下：

```
V5ClientConfig config = new V5ClientConfig(Context context);
config.setNickname("昵称");  // 设置昵称
config.setGender(1);  // 设置性别
config.setAvatar("http://static.v5kf.com/images/web/fodder/xlogo.png");  // 设置头像URL
config.setUid(String uuid); // 【必须】,设置用户ID，区分APP登录的不同账号
```

6.2.7 消息推送设置

推送参数设置：

```
V5ClientConfig config = V5ClientConfig.getInstance(Context context);

config.setDeviceToken("device_token 字符串"); // 【必须】,否则离线无法接收通知，离线消息通知发送到在 V5KF 后台配置的对应推送服务器地址，或者第三方推送平台
```

推送消息接受会收到后，在通知栏点击时需要打开客服会话界面。若您使用第三方推送平台将会返回一组自定义参数 ”v5_action” : ”new_message”，以此区分是否是来自 V5 智能客服系统的消息；若您使用自己的推送服务器，您可以在接收到消息后自行标记此消息并推送到客户端 APP（根据 device_token 识别接收客户端），然后启动您自定义的客服会话界面。

6.2.8 查询会话消息

1. 获取当前会话消息

当退出对话后，一定时间内再次进入会话界面会话并没有结束，需要获取刚刚会话的消息，通过下面接口获取：

```
V5ClientAgent.getInstance().getCurrentMessages (int offset, int size,
OnGetMessagesCallback callback)
```

其中 offset 为请求起始位置，size 为最多返回消息数，返回的 finish 为 true 时说明已没有更多会话，offset 和 size 均为 0 时表示查询当前会话全部消息。OnGetMessagesCallback 为获取历史消息的回调，以表示历史消息获取成功，参数为消息对象列表：

```
public interface OnGetMessagesCallback {
    public void complete(List<V5Message> msgs); // 执行完成

}
```

2. 获取历史会话消息

当开启消息缓存时，可以通过下面的接口查询缓存的历史消息：

```
V5ClientAgent.getInstance().getHistoricalMessages(Context    context,
int offset, int size, OnGetMessagesCallback callback)
```

参数含义同上，多了个 Context 类型参数。

此外，提供清空历史消息缓存接口：

```
V5ClientAgent.getInstance().clearLocalHistoricalMessages(Context
context);
```

6.2.9 其他设置

客户端的调试日志显示和其他配置：

```
  V5ClientConfig config = new V5ClientConfig(Context context);
  config.setShowLog(true); // 是否打印调试日志，默认为true
  config.setLogLevel(V5ClientConfig.LOG_LV_DEBUG); // 日志级别，默认为全
部显示
  // 设置是否默认转人工客服
  config. setDefaultServiceByWorker(true); // 默认为机器人服务
```

7 注意事项

7.1 代码混淆

代码混淆时需要在混淆配置文件中加入下面内容：

```
#必须
-keepattributes InnerClasses -keep class **.R$* { <fields>; }
```

```
#含腾讯地图 SDK 还需加入
-dontwarn org.eclipse.jdt.annotation.**
-keepattributes *Annotation*
-keepclassmembers class ** {
    public void on*Event(...);
}
-keepclasseswithmembernames class * {
    native <methods>;
}
```

7.2 发布提醒

应用发布时注意关闭日志打印。此外，使用不带 UI 的核心库接口开发的开发者，要注意其中 6.2.4 生命周期的处理不可以遗漏。

7.3 版本更新

SDK 存在新版本时，请尽量更新到最新版本 SDK，注意查看文档末尾的更新记录，以根据更新内容完成相应修改。

8 消息结构

8.1 异常消息

名称	说明	备注
o_error	错误码。参考值如下： 0 - 正常 50001 - 无效的方法(GET/POST/PUT) 50002 - 无效参数 50003 - 无效账号 50004 - 账号被禁止 50005 - 错误的请求域 50006 - 内部错误 50007 - 请求 URL 错误 50008 - 请求超时 50009 - 请求数据错误 50010 - 会话结束关闭 50011 - 无效会话 ID -1000 - 未初始化 SDK 或初始化失败 -1001 – 未开启消息服务 -1002 - 未获取到 url	整数

	-1003 - 站点认证失败 -1004 - 网络错误 -1005 - 服务连接断开 -1006 - 图片上传失败	
o_errmsg	错误描述。	字符串

异常消息通过 MessageListener 的 onError(V5KFException e)函数回调，错误码和对应的说明如上表。

8.2 会话消息

发送和接收消息都是 JSON 格式字符串，其中 o_type 为 message 的消息为会话消息，通过转换为消息对象 V5Message 传递，包含文本、位置、图片等等子类消息，基类 V5Message 中主要成员定义如下：

```
private int state; // 消息状态发送状态
private int hit; // 问题命中与否 0-问题未能有效回答 1-问题找到合适答案
private int message_type; // 消息类型
private int direction; // 消息标志，定义如下：
                       //  0－座席发出的消息
                       //  1－客户发出的消息
                       //  2－机器人发出的消息
                       //  7－发给座席的求助信息
                       //  8－相关问题消息
                       //  9－评价问卷
private long create_time; // 时间戳
private List<V5Message> candidate; // 相关问题内容
```

消息状态有：V5Message.*STATE_ARRIVED* 、V5Message.*STATE_FAILURE* 、V5Message.*STATE_SENDING*、V5Message.*STATE_UNKNOW* 四种。

V5Message 是所有类型消息的基类，通过“message_type”来区分消息类型，单个消息仅包含一种类型的消息内容，目前支持的消息类型示例如下。

消息内容中不同消息类型对应的成员定义如下：

消息类型	内容	说明
1	String content; // 文本内容	文本消息：V5TextMessage

2	String pic_url; // 图片 URL
	String media_id; // 媒体 ID
	String thumbnail_url; // 缩略图 URL
	String filePath; // 本地文件路径
		图片消息：V5ImageMessage
		注：本地图片上传成功后才会有图片 URL 和缩略图 URL

3	double x; // 纬度
	double y; // 经度
		位置消息：V5LocationMessage

	double scale; // 精度
	String label; // 位置描述标签	

9	List<V5ArticleBean> articles
	（其中V5ArticleBean包含：
	String title; // 标题
	String pic_url; // 图片URL
	String url; // 文章URL
	String description; // 简述）
		图文消息（多图文）：
		V5ArticlesMessage

25	int argc; // 参数个数
	String argv; // 参数内容
	int code; // 控制代码
		控制消息：
		V5ControlMessage

当接收到的消息类型为 SDK 所不支持的类型时，该类型会以 V5JSONMessage 来表示，内含一个 JSONObject 类型成员，包含接收到的完整消息内容。

会话应答消息通过 MessageListener 的 onMessage(V5Message message)函数回调，接收到的文本消息示例如下，即一个 V5TextMessage 包含的信息：

```
{
    "content": "你好！",
    "create_time": "1447323666",
    "direction": 2,
    "hit": 1,
    "message_type": 1,
   "o_type": "message"
}
```

接收文本消息的处理：

```
  public void onMessage(V5Message message) {
    if (message.getMessage_type() == V5MessageDefine.MSG_TYPE_TEXT) {
        V5TextMessage textMessage = (V5TextMessage) message;
        // 处理文本消息
         // ......
    }
  }
```

图目录

更新记录

- 2015/12/15 文档版本 Ver0.6，SDK 版本 v1.0.2

  1. 【修改】修改消息发送回调接口名称：MessageSendHandler -> MessageSendCallback。
  2. 【增加】消息接口 V5MessageListener 增加方法：onConnect()，表示与服务端连接建立成功，方可开始发送消息。
  3. 【增加】V5ClientConfig 中增加用户 uid（区分多用户账号切换情况）设置和推送设备 device_token 设置，须填写上第三方推送平台的 device_token 以识别推送终端。
  4. 【增加】增加本地图片发送功能。
  5. 【修改】修改 AndroidManifest.xml 中的 activity 和 service 的配置，取消 android:process=":v5kf"，解决因多进程中单例多个实例化导致的 V5ClientConfig 配置信息失败问题。
  6. 【增加】增加客户离线后消息推送到第三方平台，需要在 V5 官网后台做对应配置，选择对应推送平台并配置，否则离线后接收不到消息。

- 2015/12/17 文档版本 Ver0.7，SDK 版本 v1.0.3

  1. 【修改】取消 SDK 中 APP_KEY 的填写，修改了 SDK 初始化认证方式。