V2.7.0-Android-SDK对接集成文档

本文档的阅读人群为参与评估或部署智齿客服的产品经理和研发人员。文档提到的appkey等信息，请使用智齿客服超管账号登陆管理后台按照指引查询。

智齿客服SDK为企业提供了一整套完善的智能客服解决方案。智齿客服 SDK 既包含客服业务逻辑，也提供交互界面；企业只需简单两步，便可在App中集成智齿客服，让App拥有7*24小时客服服务能力。

<智齿客服SDK>

管理员可以在后台「设置-支持渠道-APP」添加APP，然后按照本接入文档说明完成SDK对接。

智齿客服SDK具有以下特性：

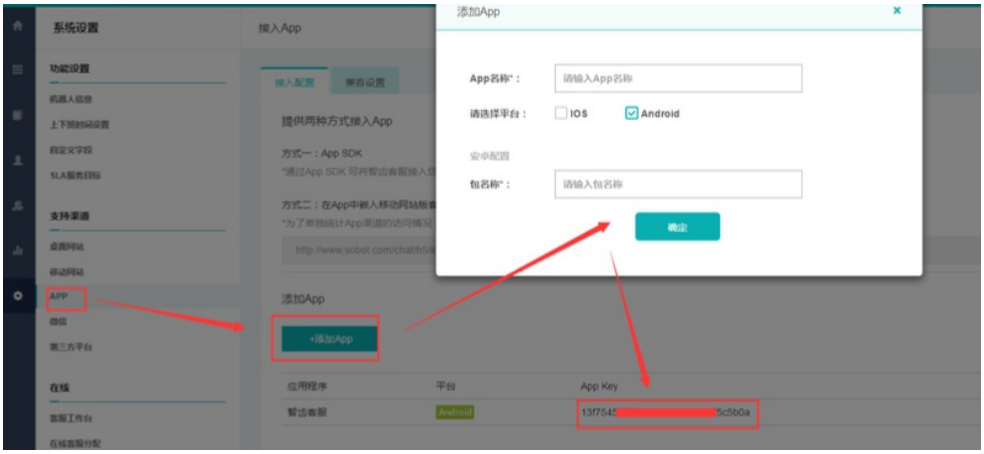
- 在线咨询：咨询机器人、咨询人工客服（收发图片、发送语音）、发送表情；
- 指定技能组接待；
- 排队或客服不在线时引导用户留言；
- 机器人优先模式下隐藏转人工按钮，N次机器人未知问题问题是显现；
- 客服满意度评价：用户主动满意度评价+用户退出时询问评价；
- 传入用户资料：用户对接ID+基础资料+自定义字段；
- 传入商品来源页：来源页标题+来源页URL；
- 高度自定义UI；

准备工作

一、注册智齿账号

请先确保已注册智齿客服账号（www.sobot.com）。

1 获取appKey



<管理后台-设置-支持渠道-APP>

二、下载智齿SDK

点击下载最新最新SDK包。

Android_SDK

快速集成

一、将SDK添加至项目（必须）

1 AndroidStudio集成(有以下两种方式，任选一种即可)

(1)远程依赖

```
dependencies {
    //以下三项必填
    //xxx为版本号请填写您要接入的版本 例如2.4.0
    compile 'com.sobot.chat:sobotsdk:xxx'
    compile 'com.squareup.okhttp3:okhttp:3.6.0'
    compile 'com.android.support:support-v4:25.3.1'
    //目前支持常见的3种图片加载库，必须在下面的图片加载库中选择一个添加依赖
    //compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
    //compile 'com.github.bumptech.glide:glide:3.7.0'
```

```
    compile 'com.squareup.picasso:picasso:2.5.2'
}
```

【注意】由于glide v3版本和v4版本的接口完全不同，因此我们为了方便您的使用，采用了特殊的集成方式使sdk可以支持任 意版本的glide。正常情况下，您使用glide 时，直接添加glide依赖和sobotsdk的依赖，sdk即 可正常使用。如果报错，那么把glide 升级到4.4.0版本以上即可。
并且参照混淆文件（Android_SDK_x.x.x\AndoridStudio\Demo\SobotSDK\sobotsdkdemo\proguard-rules.pro）中的混淆配置添加混淆规则。

(2)导入Module
解压下载的智齿Android_SDK_XXX.zip文件，将Android_SDK_XXX\SobotSDK\androidstudio\\sobotsdk文件导入您的项目中，操作方法 为:File-->New-->Import Module。

- 添加项目依赖

将lib库添加到项目依赖、操作方法为:ctrl+alt+shift+s.在弹出的对话框中选择您的Module。选中右侧选项卡Dependencies，点击“+”选择Module dependency，在弹出的对话框中选择选中sobotsdk，点击“ok”。Build-->clean project。
完成上述步骤之后build.gradle中如下所示:

```
dependencies {
    //以下三项必填
    compile project(':sobotsdk')
    compile 'com.squareup.okhttp3:okhttp:3.6.0'
    compile 'com.android.support:support-v4:25.3.1'
    //目前支持常见的3种图片加载库，必须在下面的图片加载库中选择一个添加依赖
    //compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
    //compile 'com.github.bumptech.glide:glide:3.7.0'
    compile 'com.squareup.picasso:picasso:2.5.2'
}
```

- 安卓6.0权限适配

如果您的项目需要适配6.0权限，将您项目中的targetSdkVersion修改为23以上即可。

二、初始化方法 （必须）

1、在Application的onCreate方法中调用初始化方法。

注意：从SDK-2.4.3版本开始，必须调用以下方法，之前版本可以忽略本项。

```
/**
* 初始化sdk
* @param context 上下文 必填
* @param appkey 用户的appkey 必填 如果是平台版用户需要传总公司的appkey
* @param uid   用户的唯一标识，不能传一样的值，可以为空
*/
SobotApi.initSobotSDK(context, "Your appkey", "");
```

2、在客服入口按钮的响应函数中加入初始化方法（必须）

```
Information info = new Information();
info.setAppkey("Your AppKey"); //分配给App的的密钥
/**
```

```
* @param context 上下文对象
* @param information 初始化参数
*/
SobotApi.startSobotChat(context, information);
```

完成以上步骤的对接后，用户则可通过App中的客服入口进入智齿客服的服务页面了。
当然，为了满足客户个性化的需要，智齿SDK提供了强大的UI自定义能力，支持对接登陆用户的身份资料，在特定服务场景下设置特定的客服方案。
注意：如果发生应用打开后自动回退到上一界面，请检查appkey是否正确，另外，sdk2.0版本以后不需要使用sysnum,应该重新获取appkey（获取方法参考准备工作中的1.获取appkey），将新应用的appkey传入。

如有特殊需求，SDK 还提供了以 fragment 嵌入的方式集成会话界面，开发者可以更灵活的使用 SDK。示例代码如下（也可参考SobotChatActivity中的实现)

```
SobotChatFragment fragment = SobotChatFragment.newInstance(informationBundle);
FragmentManager fm = getSupportFragmentManager();
FragmentTransaction transaction = fm.beginTransaction();
// containerId 为 ViewGroup 的 resId
transaction.replace(containerId, fragment);
try {
   transaction.commitAllowingStateLoss();
} catch (Exception e) {
}
```

初始化SDK（同时设定客服配置）

一、对接用户资料

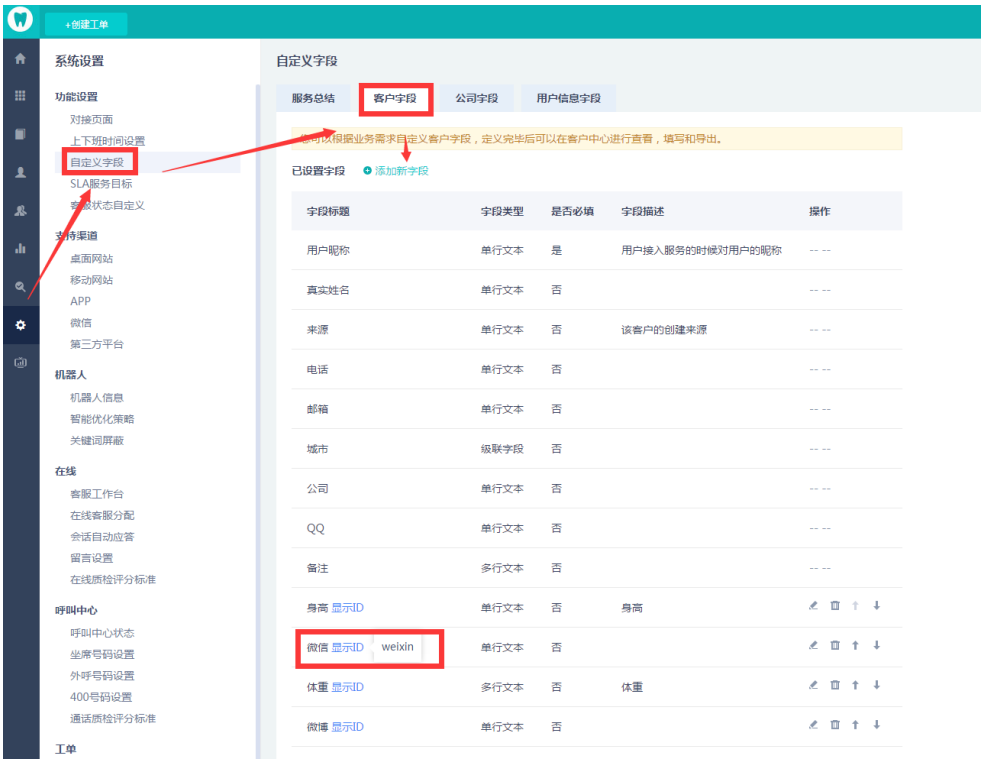
开发者可以直接传入这些用户信息，供客服查看。
注意：uid为用户唯一标识，不能传入一样的值，如果传入值为""空串，那么会获取手机设备id作为uid

```
Information info = new Information();
//用户编号
//注意：uid为用户唯一标识，不能传入一样的值
info.setUid("");
//用户昵称，选填
info.setUname("");
//用户姓名，选填
info.setRealname("");
//用户电话，选填
info.setTel("");
//用户邮箱，选填
info.setEmail("");
//自定义头像，选填
info.setFace("");
//用户QQ，选填
info.setQq("");
```

```
//用户备注，选填
info.setRemark("");
//访问着陆页标题，选填
info.setVisitTitle("");
//访问着陆页链接地址，选填
info.setVisitUrl("");
Map<String,String> customInfo = new HashMap<String, String>();
customInfo.put("your key", "your value");
....
//自定义用户资料
info.setCustomInfo(customInfo);
```

设置用户自定义字段

如果上述设置的用户资料不能满足您的需求，那么您可以在工作台自行配置所需要显示的字段，配置方法如下图:



设置好自定义的字段后，将自定义字段的id作为key以下面的形式传入:

注意：自定义字段在console页面添加新字段以后会在下面生成对应的字段ID。点击显示ID，就可以显示自定义key.添加以后可以删除。删除以后再添加一次的时候，这个key就会发生变化。需要谨慎操作。

```
//设置用户自定义字段
Map<String,String> customerFields = new HashMap<>();
customerFields.put("weixin","your wechat");
customerFields.put("weibo","your weibo");
customerFields.put("sex","女");
customerFields.put("birthday","2017-05-17");
info.setCustomerFields(customerFields);
```

二、自定义聊天页面标题样式

2.1 自定义标题栏背景

```
//设置标题栏的背景图片，选填
info.setTitleImgId(R.drawable.sobot_delete_hismsg_normal);
```

2.2 自定义标题栏显示文案

```
/**
 * 设置聊天界面标题显示模式
 * @param context 上下文对象
 * @param mode titile的显示模式
 *          SobotChatTitleDisplayMode.Default:显示客服昵称(默认)
 *          SobotChatTitleDisplayMode.ShowFixedText:显示固定文本
 *          SobotChatTitleDisplayMode.ShowCompanyName:显示console设置的企业名称
 * @param content 如果需要显示固定文本，需要传入此参数，其他模式可以不传
*/
SobotApi.setChatTitleDisplayMode(context,SobotChatTitleDisplayMode.Default,"");
```

三、自定义接入模式

根据自身业务的需求，可进行以下初始化参数配置，控制接入模式：

```
//默认false：显示转人工按钮。true：智能转人工
info.setArtificialIntelligence(false);
//当未知问题或者向导问题显示超过(X)次时，显示转人工按钮。
//注意：只有ArtificialIntelligence参数为true时起作用
info.setArtificialIntelligenceNum(X);
//是否使用语音功能 true使用 false不使用  默认为true
info.setUseVoice(true);
//是否使用机器人语音功能 true使用 false不使用 默认为false,需要付费才可以使用
info.setUseRobotVoice(false);
//客服模式控制 -1不控制 按照服务器后台设置的模式运行
//1仅机器人 2仅人工 3机器人优先 4人工优先
info.setInitModeType(-1);
//设置机器人模式下输入关键字转人工
HashSet<String> tmpSet = new HashSet<>();
tmpSet.add("转人工");
tmpSet.add("人工");
info.setTransferKeyWord(tmpSet);
```

四、自定义客户转入的技能组

获取技能组ID：

企业可通过此配置实现在特定场景下，对特定用户群体以特定的客服模式接待。

```
//预设技能组编号
info.setSkillSetId("your skillCode");
//预设技能组名称，选填
info.setSkillSetName("your skillName");
```

五、发送商品页信息

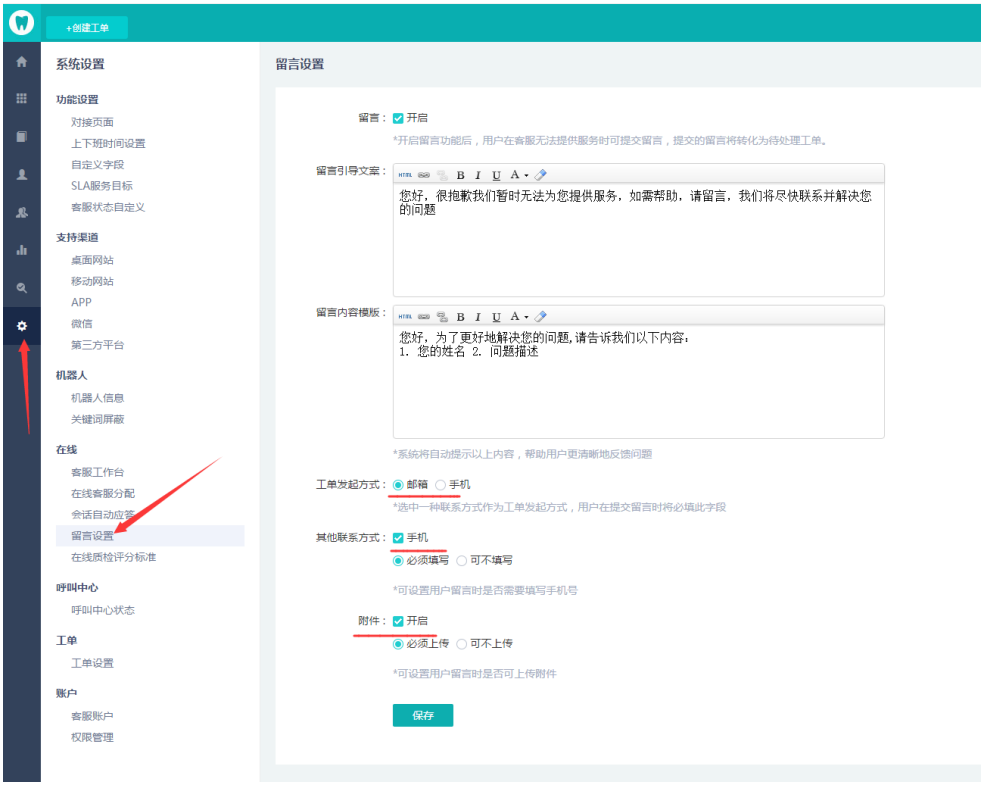
在用户与客服对话时，经常需要将如咨询商品或订单发送给客服以便客服查看。咨询对象目前最多支持发送5个属性(title,imgUrl,fromUrl,describe,lable)，其中(title,fromUrl)为必填字段，如下以商品举例说明：

```
//咨询内容
ConsultingContent consultingContent = new ConsultingContent();
//咨询内容标题，必填
consultingContent.setSobotGoodsTitle("XXX超级电视50英寸2D智能LED黑色");
//咨询内容图片，选填 但必须是图片地址
consultingContent.setSobotGoodsImgUrl("http://www.li7.jpg");
//咨询来源页，必填
consultingContent.setSobotGoodsFromUrl("www.sobot.com");
//描述，选填
consultingContent.setSobotGoodsDescribe("XXX超级电视 S5");
//标签，选填
consultingContent.setSobotGoodsLable("￥2150");
//可以设置为null
info.setConsultingContent(consultingContent);
```

六、设置留言提交校验逻辑

注意：在sdk2.1之前(包括2.1)，留言提交校验逻辑为代码中本地配置，自sdk2.2版本起，此项配置修改为pc工作台配置，请旧版本用户升级时做好相应的设置。

留言中的邮箱、电话、附件这三个参数的校验和显示逻辑可在pc端console页面配置。



留言中提交的昵称字段为选填项。

注意：由于昵称字段在pc工作台没有相应的设置项，但是考虑到旧版本sdk依旧有使用这个配置项的用户，因此此项配置依旧使用本地代码设置。

设置昵称字段是否显示：

```
info.setShowNikeNameTv(flag);//true 表示显示 false 表示不显示
```

昵称设置为显示时，可设置昵称字段是否为必填字段：

```
info.setShowNikeName(boolean);//true 表示必填 false 表示选填
```

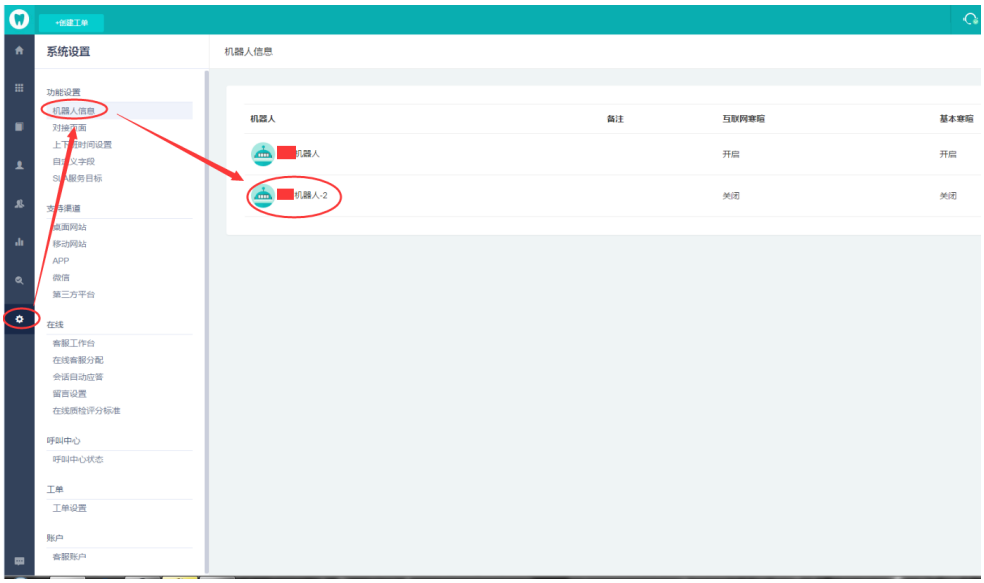
七、返回时要求用户评价

开启后，当用户从SDK会话页面中返回至App时，且用户达到评价标准却未评价时，给用户推送评价，可提升用户的评价率，具体配置如下：

```
//返回时是否弹出满意度评价
info.setShowSatisfaction(true);
```

八、自定义客户转入指定的机器人

选择机器人：



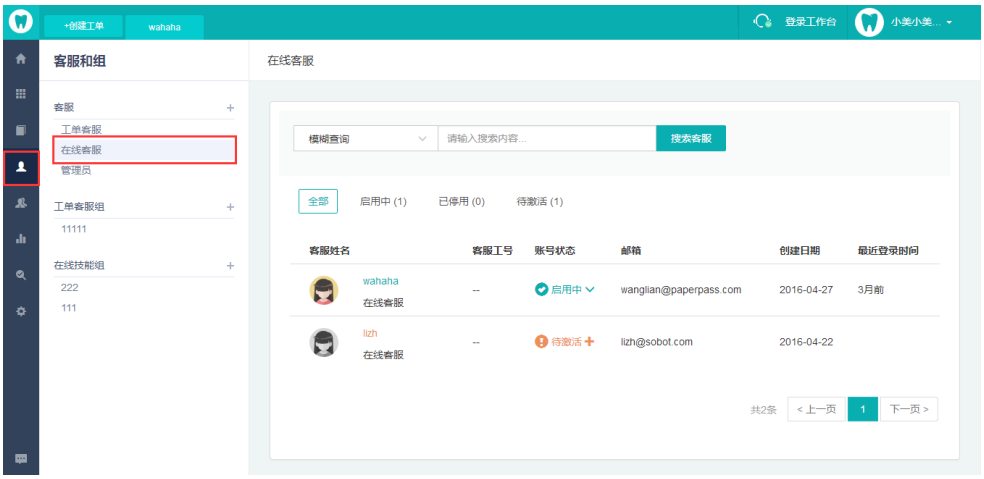
获取机器人编号：

企业可通过此配置实现在特定场景下，对特定用户群体以特定的机器人接待。

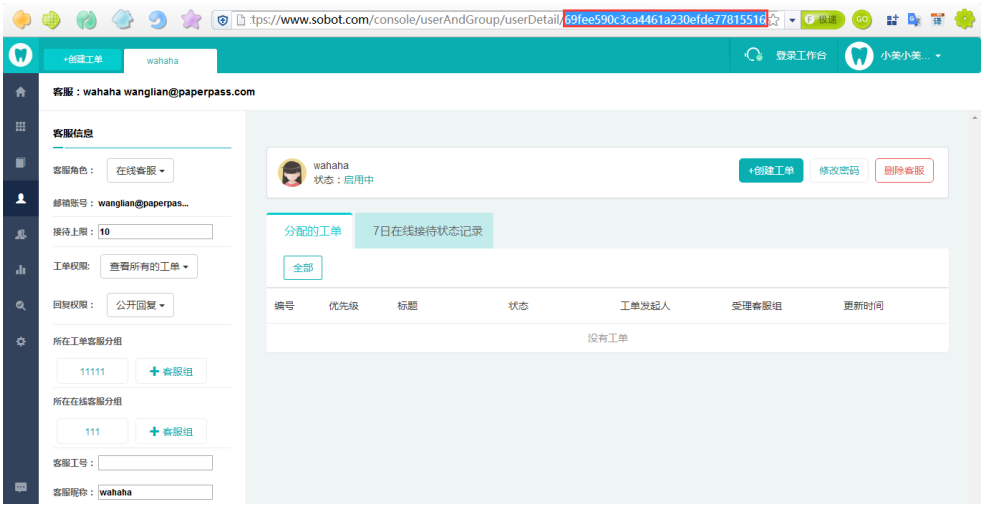
```
//设置机器人编号

info.setRobotCode("your robot code");
```

九、自定义客户转入指定的客服

选择客服：



获取客服ID:



企业可通过此配置实现在特定场景下，对特定用户群体以特定的客服接待。

```
//转接类型(0-可转入其他客服，1-必须转入指定客服)

info.setTranReceptionistFlag(1);

//指定客服id

info.setReceptionistId("your Customer service id");
```

十、设置转接成功后自动发消息

sdk可以设置转接成功后自动发消息

```
//设置发送模式

//SobotAutoSendMsgMode.Default 默认 不发送

//SobotAutoSendMsgMode.SendToRobot 只给机器人发送

//SobotAutoSendMsgMode.SendToOperator 只给人工客服发送

//SobotAutoSendMsgMode.SendToAll 全部发送

info.setAutoSendMsgMode(SobotAutoSendMsgMode.SendToAll.setContent("your msg"));
```

API说明

一、启动聊天界面

```
/**
```

```
* @param context 上下文对象
* @param information 初始化参数
*/
SobotApi.startSobotChat(context, information);
```

二、注册广播、获取新收到的信息和未读消息数

注册广播后，当消息通道连通时，可以获取到新接收到的消息。

1 注册广播

```
/**
* action:ZhiChiConstants.sobot_unreadCountBrocast
*/
IntentFilter filter = new IntentFilter();
filter.addAction(ZhiChiConstant.sobot_unreadCountBrocast);
contex.registerReceiver(receiver, filter);
```

2 接收新信息和未读消息数

在BroadcastReceiver的onReceive方法中接收信息。

```
public class MyReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    int noReadNum = intent.getIntExtra("noReadCount", 0);
    String content = intent.getStringExtra("content"); //未读消息数
    unread_msg_num.setText(noReadNum + "");
    //新消息内容
    LogUtils.i("新消息内容:"+content);
  }
}
```

三、获取未读消息数

当用户不处在聊天界面时，收到客服的消息会将未读消息数保存在本地，如果需要获取本地保存的未读消息数，那么在需要的地方调用该方法即可。如下：

```
/**
* @param context 上下文对象
* @return int
*/
SobotApi.getUnreadMsg(context);
```

四、设置是否开启消息提醒

当用户不处在聊天界面时，收到客服的消息，APP 可以在通知栏或者聊天入口给出提醒。通知栏提醒可以显示最近一条消息的内容，并提供给用户快速进入 APP 的入口。

```
/**
* 设置是否开启消息提醒 默认不提醒
* @param context
* @param flag
```

```
* @param smallIcon 小图标的id 设置通知栏中的小图片，尺寸一般建议在24×24
* @param largeIcon 大图标的id
*/
public static void setNotificationFlag(Context context,boolean flag,int smallIcon,int largeIcon);
```

用户点击通知栏会发出广播，监听广播实现跳转到指定Activity

action:

常量：ZhiChiConstant.SOBOT_NOTIFICATION_CLICK

或者字符串：“sobot_notification_click”

五、自定义超链接的点击事件

如想自定义聊天内容中超链接的点击事件，需要使用以下监听方式：

```
SobotApi.setHyperlinkListener(new HyperlinkListener() {
  @Override
  public void onUrlClick(String url) {
    LogUtils.i("点击了超链接，url="+url);
  }
  @Override
  public void onEmailClick(String email) {
    LogUtils.i("点击了邮件，email="+email);
  }
  @Override
  public void onPhoneClick(String phone) {
    LogUtils.i("点击了电话，phone="+phone);
  }
});
```

六、自定义聊天记录显示的时间范围

如想设置用户只能看到xx天内的聊天记录，那么可以调用以下方法进行设置

```
/**
 * 控制显示历史聊天记录的时间范围
 * @param time  查询时间(例:100-表示从现在起前100分钟的会话)
*/
SobotApi.hideHistoryMsg(context,time);
```

七、配置用户提交人工满意度评价后释放会话

```
/**
 * 配置用户提交人工满意度评价后释放会话
 * @param context 上下文对象
 * @param flag true 表示释放会话  false  表示不释放会话
 */
SobotApi.setEvaluationCompletedExit(context,flag);
```

八、设置跳转到留言页的监听

sdk中留言可跳转到自定义页面，如有此需求，可以使用如下方法进行设置：

```
SobotApi.setSobotLeaveMsgListener(new SobotLeaveMsgListener() {
    @Override
    public void onLeaveMsg() {
      ToastUtil.showToast(getApplicationContext(),"在这里实现方法，跳转页面");
    }
});
```

九、自定义自动应答语

sdk中的自动应答语可以在pc工作台进行动态设置，如果pc工作台的设置满足不了您的需求，那么您可以使用以下接口在代码中进行本地配置

```
SobotApi.setCustomAdminHelloWord(context,"自定义客服欢迎语");
SobotApi.setCustomRobotHelloWord(context,"自定义机器人欢迎语");
SobotApi.setCustomUserTipWord(context,"自定义用户超时提示语");
SobotApi.setCustomAdminTipWord(context,"自定义客服超时提示语");
SobotApi.setCustomAdminNonelineTitle(context," 自定义客服不在线的说辞");
SobotApi.setCustomUserOutWord(context," 自定义用户超时下线提示语");
```

十、“+”号面板菜单扩展

客服聊天界面中点击“+”按钮后所出现的菜单面板，可以根据需求自行添加菜单，代码如下：

```
 private void customMenu(){
      //添加扩展菜单数据
      ArrayList<ChattingPanelUploadView.SobotPlusEntity> objects = new ArrayList<>();
      /**
       * SobotPlusEntity为自定义菜单实体类
       * @param iconResId 菜单图标 drawableId
       * @param name     菜单名称
       * @param action    菜单动作 当点击按钮时会将对应action返回给callback
       *              以此作为依据，判断用户点击了哪个按钮
       */

      objects.add(new
ChattingPanelUploadView.SobotPlusEntity(R.drawable.sobot_camera_picture_button_selector,
"位置", "action_location"));
      objects.add(new
ChattingPanelUploadView.SobotPlusEntity(R.drawable.sobot_camera_picture_button_selector,
"签到", "action_sing_in"));
      objects.add(new
ChattingPanelUploadView.SobotPlusEntity(R.drawable.sobot_camera_picture_button_selector,
"收藏", "action_ollection"));

      //添加数据
      SobotUIConfig.pulsMenu.menus = objects;
      //设置回调
      SobotUIConfig.pulsMenu.sSobotPlusMenuListener = new SobotPlusMenuListener() {
        @Override
        public void onClick(View view, String action) {
```

```
            //action与实体类中的action对应
            ToastUtil.showToast(getApplicationContext(), "action:"+action);
        }
    };
}
```

十一、发送位置信息

如您的app需要发送客户的位置信息，请参照以下步骤设置(其中地图定位页面需要开发者自行开发)：

1、客服聊天界面配置位置发送按钮（显示在点击“+”按钮的菜单面板中，只在转人工后显示），代码如下：

```
//菜单动作 当点击按钮时会将对应action返回给callback以此作为依据，
//判断用户点击了哪个按钮，可自行定义
final String ACTION_LOCATION = "sobot_action_location";
//配置位置发送按钮
ChattingPanelUploadView.SobotPlusEntity locationEntity = new
ChattingPanelUploadView.SobotPlusEntity(R.drawable.sobot_location_btn_selector,
ResourceUtils.getResString(getApplicationContext(), "sobot_location"), ACTION_LOCATION);
List<ChattingPanelUploadView.SobotPlusEntity> tmpList = new ArrayList<>();
tmpList.add(locationEntity);
SobotUIConfig.pulsMenu.operatorMenus = tmpList;
```

2、设置位置发送按钮的回调：

```
SobotUIConfig.pulsMenu.sSobotPlusMenuListener = new SobotPlusMenuListener() {
    @Override
    public void onClick(View view, String action) {
        if (ACTION_LOCATION.equals(action)) {
            //点击了位置按钮
            Context context = view.getContext();
            //todo 点击位置按钮后打开定位页面(此页面需要开发者自行集成地图sdk 完成地图定位页面)
            //context.startActivity(new Intent(context,LocationActivity.class));
        }
    }
};
```

3、在地图定位页面获取位置信息后发送给客服：

```
SobotLocationModel locationData = new SobotLocationModel();
//地图快照，必须传入本地图片地址，注意：如果本地图片地址不存在会显示发送失败
locationData.setSnapshot(context.getCacheDir()+File.separator+"tmp_pic.jpg");
//纬度
locationData.setLat("40.001630");
//经度
locationData.setLng("116.353313");
//标点名称
locationData.setLocalName("云景四季餐厅");
//标点地址
```

```
locationData.setLocalLabel("学清路38号金码大厦A座23层");


SobotApi.sendLocation(context, locationData);
```

十二、注销

用户在应用中退出登陆时需要调用 SDK 的注销操作（只在切换账号时调用），该操作会通知服务器进行推送信息的解绑，避免用户已退出但推送依然发送到当前设备的情况发生。当用于用户退出登录时调用以下方法：

注意：调用此方法会造成通道连接断开，此时用户将无法收到消息。

```
/**
* @param context 上下文对象
*/
SobotApi.exitSobotChat(context);
```

接口说明

本项的阅读对象为：需要完全自主开发UI的客户

一、初始化接口：sobotInit

初始化一些必要的参数，本接口成功调用一次即可。

sobotInit(Information info, StringResultCallBack<ZhiChiInitModel> resultCallBack);

请求参数说明：

变量名	数据类型	说明
info	Information	调用初始化接口之前必须设置此对象的相关属性
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
    "code": 1,
    "data": {
        "uid": "6ea1efaad73246d2bd778e65c9a69358867993024100230",
        "robotLogo": "https://img.sobot.com/console/common/face/robot.png",
        "adminHelloWord": "您好，请问有什么可以帮您的？ ",
        "manualCommentTitle": "服务态度差,回答不及时,没解决问题,不礼貌",
        "msgTmp": "您好，为了解决您的问题, ",
        "type": 2,
        "isblack": 0,
        "robotUnknownWord": "非常对不起，不知道怎么回答这个问题呢，我会努力学习的。\n",
        "groupflag": 0,
        "guideFlag": 0,
        "msgTxt": "您好，很抱歉我们暂时无法为您提供服务，如需帮助，请留言\n",
        "onORoff": 3,
        "msgFlag": 0,
        "adminTipTime": 1,
        "adminNonelineTitle": "抱歉，暂无人工客服在线",
        "userOutWord": "由于很久没有收到您的消息，系统自动18513666499结束了本次会话。",
        "userTipTime": 3,
        "userTipWord": "您在思考人生？有问题请随时提问哦111111~\n",
```

```
        "robotHelloWord": "您好，很高兴为您服务，请问有什么可以帮您的？ \n",
        "inputTime": 2,
        "ustatus": 0,
        "adminTipWord": "抱歉让您久等，客服妹子已经忙翻了，请稍候。\n",
        "companyName": "zhichi",
        "cid": "341c9caf6bfe43419b7a5cde363c3514",
        "robotName": "zhichi机器人",
        "companyStatus": 0,
        "userOutTime": 3,
        "color": "#09aeb0",
        "companyId": "6ea1********46d2bd778e********58",
        "robotCommentTitle": "答非所问,理解能力差,问题不能回答,不礼貌"
        "uname": "北京用户",
        "face": "",
    },
    "msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	String	请求结果码：成功code==1 失败 code==0
uid	String	用户id
cid	String	会话id
robotLogo	String	机器人头像
adminHelloWord	String	客服欢迎语
manualCommentTitie	String	评价标签，以逗号分隔
msgTmp	String	留言内容模板
type	int	接入方式 1只有机器人 2 仅人工 3.智能客服-机器人优先 4智能客服-人工客服优先
isblack	int	是否是黑名单 0否1是
robotUnknownWord	String	机器人未知措辞
groupflag	String	是否开启技能组 0不分组，1分组
guideFlag	int	机器人常见问题引导欢迎语的开关 1开启 0关闭
msgTxt	String	留言引导文案
msgFlag	int	留言开关1关闭0开启
adminTipTime	int	客服超时提示时间
adminTipWord	String	客服超时提示文案
adminNonelineTitle	String	无客服在线文案
userOutWord	String	用户超时下线文案
userTipTime	int	用户超时提示时间
userTipWord	String	用户超时提示文案
userOutTime	int	用户超时下线时间
robotHelloWord	String	机器人欢迎语
inputTime	int	发送正在输入的频率
ustatus	String	用户当前状态: -2.排队中 -1.机器人 0.离线 1.在线
companyName	Stirng	企业名称
robotName	String	机器人名称
companyStatus	int	公司付费状态:-1.免费 0.试用 10.都付费 11.机器人付费 12.人工客服付费 2.永久
color	String	颜色
companyId	String	公司id
robotCommentTitle	String	机器人的评价标签，用逗号分隔

二、查询cid接口：queryCids

本接口调用后可以查询到用户的cid列表，获取cid列表后，可以使用cid调用getChatDetailByCid接口进行历史纪录的查询。

queryCids(String uid,long time, StringResultCallBack<ZhiChiCidsModelResult> resultCallBack);

请求参数说明:

变量名	数据类型	说明
uid	必填	用户id
time	选填	查询时间(例:100 表示从现在起前100分钟的会话)，默认为0
resultCallBack	StringResultCallBack	回调

响应示例:

```
{
   "code": 1,
```

```
    "data": {
        "cids": [
            "d7e416f161af49afb856236db88e1fe1",
            "73c4a12d89fc4f13b89cd02e0583e22f",
            "0bf1f68262a54bd294a18f04762f057d",
        ]
    },
    "msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
cids	list	cid列表数据,用于历史记录查询

三、获取聊天记录接口：getChatDetailByCid

sdk中所有聊天纪录都是以cid为单位查询得出，如果想获取用户的聊天记录，那么首先需要调用queryCids接口获取cid列表后，再使用cid进行聊天记录的查询。

getChatDetailByCid(String uid, String cid,
StringResultCallBack<ZhiChiHistoryMessage> resultCallBack);

请求参数说明：

变量名	数据类型	说明
uid	String	初始化接口返回的用户id
cid	String	会话id（由queryCids接口查出cid列表后，可以获取历史会话的聊天纪录）
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
"code": 1,
"data": [
            {
        "date": "2017-03-01",
        "content": [
                {
                    "cid": "846ac1054e5c46cca9887f88e862d6a2",
                    "action": 5,
                    "type": 5,
                    "sender": "9e74cbf3b9564ccca28511d45a5abdbe",
                    "senderName": "jin",
                    "senderType": 2,
                    "senderFace": "https://img.sobot.com/cae967f.jpg",
                    "t": "1488362243542",
                    "ts": "2017-03-01 17:57:23",
                    "msgType": 0,
                    "msg": "111",
                    "sdkMsg": {
                                "sugguestions": null,
                                "answerType": null,
                                "stripe": null,
```

```
                                "question": "",
                                "answer": {
                                        "msgType": 0,
                                        "msg": "111",
                                        "duration": null,
                                        "richpricurl": null,
                                        "richmoreurl": null
                                    }
                            },
                        "receiver": "6ea1efaad73024100230",
                        "receiverName": "北京用户",
                        "receiverType": 0,
                        "offlineType": null,
                        "msgId": "da28dc32981049ba8870204d3cae7b1c",
                        "receiverFace": ""
                        },
                ]
        }
],
"msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	String	请求结果码：成功code==1 失败 code==0
date	String	会话时间
content	String	聊天内容
cid	String	会话id
action	int	会话类型
sender	String	用户id
senderName	String	用户姓名
senderType	String	发送类型
senderFace	String	用户头像
t	Long	发送时的时间戳
ts	String	发送时间
msg	String	消息
sdkMsg	Map	消息体
answer	Map	富媒体消息 msgType 0文本 1图片 2音频 4 富文本中有图片5 富文本中纯文字 6 富文本中有视频 msg消息内容 duration语音时长 richpricurl图片链接 richmoreurl点击更多链接
suggestionList	list	引导说辞带docId
suggestions	String[]	引导说辞不带docId
answerType	int	1 直接回答，2 理解回答，3 不能回答, 4引导回答，6互联网寒暄，7 私有寒暄（包括第三方天气、快递接口）,8百科, 9 向导回答,10 业务接口
stripe	String	引导建议
receiver	String	客服id
receiverName	String	客服姓名
offlineType	int	下线类型
receiverFace	String	客服头像
receiverType	int	回复类型

四、机器人问答接口：chatSendMsgToRoot

使用本接口可以与机器人进行问答咨询，sdk中所有与机器人的聊天信息都会调用此接口。
chatSendMsgToRoot(String uid, String cid,String robotFlag,String requestText, int lanFlag, int questionFlag,String question, StringResultCallBack<ZhiChiMessage> resultCallBack);
请求参数说明：

变量名	数据类型	说明
uid	String（必填）	用户id
cid	String（必填）	当前会话id
robotFlag	String	机器人id
requestText	String	问题
questionFlag	int	问题类型0直接回答，1回答引导问题
question	String	问题内容，回答引导问题时需传入引导问题的docID
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
    "code": 1,
    "data": {
        "ustatus": -1,
        "answer": {
            "msgType": 4,
            "msg": null,
            "duration": null,
            "richpricurl": null,
            "richmoreurl": null
        },
        "answerType": "4",
        "stripe": "您问的是否是以下问题,点击或回复序号即可得到对应问题的答案：",
        "suggestionList": [
            {
                "question": "富文本",
                "docId": "27e52a49451a4e87985ae45c354fe2d9",
                "answer": "<p>富文本富文本富"
            },
            {
                "question": "富文本 两张图片富文本 两张图片富文本 两张图片富文本 两张图片",
                "docId": "0b2a12079b8e4347872d5547e59a5cac",
                "answer": "<p>富文本富文"
            }
        ],
        "question": null,
        "time": null
    },
"msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	String	请求结果码：成功code==1 失败 code==0
ustatus	int	用户状态1在线 0离线
answer	String	富媒体消息 msgType 0文本 1图片 2音频 4 富文本中有图片 5 富文本中纯文字 6 富文本中有视频 msg消息内容 duration语音时长 richpricurl图片链接 richmoreurl点击更多链接
answerType	int	1 直接回答，2 理解回答，3 不能回答, 4引导回答，6互联网寒暄，7 私有寒暄（包括第三方天气、快递接口）,8百科, 9 向导回答,10 业务接口
stripe	String	引导建议
suggestionList	list	引导说辞带docId
suggestions	String[]	引导说辞不带docId
question	String	问题内容
docId	String	标题id
answer	String	答案

五、转人工接口：connnect

sdk中调用初始化后默认为机器人问答状态，如果想连接客服进行人工问答，那么可以调用此接口，本接口成功调用一次即可。

connnect(String uid, String cid, String chooseAdminId,int tranFlag, String groupId,String groupName, boolean currentFlag, StringResultCallBack <ZhiChiMessage> esultCallBack);

请求参数说明：

变量名	数据类型	说明
uid	String	用户id
cid	String	会话id
chooseAdminId	String	指定转入的客服id（可以为空）
tranFlag	int	转接类型(0-可转入其他客服，1-必须转入指定客服)
groupId	String	技能组id（技能组ID来源：1.初始化用户配置的技能组id 2.点击技能组弹框中的某个技能组）
groupName	String	技能组名称
currentFlag	boolean	第二次以后点击转人工传true
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
   "code": 1,
   "data": {
      "count": 1,
      "aface": "https://img.sobot.com/chatr*****350b0.jpg",
      "status": 1,
      "wslink.default": "**.2**.**1.**4:6***",
      "aid": "1b6d*******4db99c3c******bd3332",
      "aname": "sobotsnbgghh",
      "wslink.bak": [
         "5*.2**.2**.1**:***5"
```

```
        ],
    },
    "msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	String	请求结果码：成功code==1 失败 code==0
status	int	转人工状态 0 排队中 1成功 2没有客服在线 3用户被拉黑 4 已经转人工 5机器人模式下已经超时下线时转人工 6如果设置指定客服的id。并且设置不是必须转入,返回6说明指定的客服不在线继续转其他客服 7转人工排队达到最大值的时候，返回 status 为 7
count	int	当前排队人数
aface	String	客服的头像
aid	String	客服id
aname	String	客服昵称
wslink.default	String	默认通道IP
wslink.bak	String	备用通道IP

六、与客服聊天接口：sendMsgToCoutom

在调用转人工接口成功后，与人工的问答消息都需要调用此接口，sdk中所有与人工的会话都会使用此接口。

sendMsgToCoutom(String content, String uid, String cid，StringResultCallBack <CommonModelBase> resultCallBack);;

请求参数说明:

变量名	数据类型	说明
uid	String	用户id
cid	String	会话id
content	String	文字消息内容
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
    "code": 1,
    "data": {
        "status": 1
    }
"msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
status	int	1发送成功 2 发送失败，表示用户已经下线

七、用户给客服发送文件接口：sendFile

在调用转人工接口成功后，发送语音、图片等文件需要调用此接口。

sendFile(String cid, String uid, String filePath, String duration, ResultCallBack<ZhiChiMessage> resultCallBack);
请求参数说明：

变量名	数据类型	说明
uid	String	用户
cid	String	会话
filePath	String	文件路径（图片和语音）
duration	String	语音的时长
resultCallBack	ResultCallBack	回调

响应示例：

```
{
    "code": 1,
    "data": {
        "status": 1,
        "url": "https://img.sobot.com/chatres/e6fb177060b91b2db544.wav"
    },
    "msg": null
}
```

响应示例说明：

字段名	数据类型	说明
code	String	请求结果码：成功code==1 失败 code==0
status	int	1 发送成功 2 发送失败
url	String	文件的网络地址

八、满意度评价接口：comment

用户对机器人或者人工进行评价的接口，系统默认一次会话只能评价一次机器人和人工，评价后信息会计入统计列表。

comment(String cid, String uid, String type, String source, String problem, String suggest, int isresolve, int commentType, StringResultCallBack <CommonModel> ResultCallBack);
请求参数说明：

变量名	数据类型	说明
uid	String	用户id
cid	String	会话id
type	String	评价类型 0机器人 1人工
source	String	评价分数 1 - 5分
problem	String	问题编号，可以为空
suggest	String	建议
isresolve	int	是否解决问题:0解决 1未解决
commentType	int	评价类型 主动评价1 邀请评价0
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
    "code": 1,
    "data": {
        "status": 1
    },
    "msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
status	int	发送成功

九、获取技能组接口：getGroupList

调用转人工接口时，如果用户打开技能组开关，并且设置了技能组，那么可以调用此接口来获取技能组。

getGroupList(String appkey,String uid, StringResultCallBack<ZhiChiGroup> resultCallBack);

请求参数说明：

变量名	数据类型	说明
appkey	String	Appkey
uid	String	用户id
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
    "code": 1,
    "data": [
        {
            "groupId": "c30ea56d46994b1093dc20ad65785d2a",
            "channelType": "5",
            "groupName": "aaa",
            "companyId": "6ea1efaad73246d2bd778e65c9a69358",
            "recGroupName": "aaa",
            "appkey": "13f7545fa0834debbe3f488daf5c5b0a",
            "isOnline": true
        },
        {
            "groupId": "ff0333886e76470db66bee3f2519063b",
            "channelType": "5",
            "groupName": "343434",
            "companyId": "6ea1efaad73246d2bd778e65c9a69358",
            "recGroupName": "343434",
            "appkey": "13f7545fa0834debbe3f488daf5c5b0a",
            "isOnline": true
        }
    ],
    "msg": {
```

```
        "ustatus": 1
    }
}
```

响应示例说明：

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
ustatus	int	1客户在线 0客户下线
groupId:	String	技能组id
channelType:	String	通道类型
groupName:	String	技能组名称
companyId:	String	企业id
recGroupName;	String	分组别名（供展示）
appkey	String	Appkey
isOnline:	String	当前技能组是否有客服在线

十、正在输入接口：input

调用转人工接口成功后，可以使用此接口来捕捉用户正在输入的消息。

input(String uid,String content, StringResultCallBack<CommonModel> resultCallBack);
请求参数说明：

变量名	数据类型	说明
uid	String	用户id
content	String	输入的内容
resultCallBack	StringResultCallBack	回调

响应示例：

```
{
    "code": 1,
    "data": {
        "status": 1
    },
    "msg": null
}
```

响应示例说明：

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
status	int	1 发送成功 0发送失败

十一、留言接口：postMsg

当用户需要进行留言时，可使用此接口进行留言提交，留言提交后会自动生成工单。

postMsg(String uid, String ticketContent, String customerEmail, String customerPhone, String companyId, String customerNick, String fileStr, StringResultCallBack<CommonModel> resultCallBack);
请求参数说明：

变量名	数据类型	说明
uid	String	用户id
ticketContent	String	留言内容
customerEmail	String	留言人邮箱
customerPhone	String	留言人电话
companyId	String	企业
fileStr	String	图片路径以逗号分隔
resultCallBack	StringResultCallBack	回调

响应示例:

```
{
    "code": 1,
    "data": {
        "status": 1
    },
    "msg": null
}
```

响应示例说明:

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
status	int	1 留言成功 0留言失败
msg	String	当status返回0是，msg有值

十二、删除聊天记录：deleteHisMsg

调用此接口后，sdk端将不再显示之前的聊天记录，但是客服工作台依旧可以查看到之前的聊天记录。

deleteHisMsg(String uid, StringResultCallBack<CommonModel> resultCallBack);

请求参数说明:

变量名	数据类型	说明
uid	String	Appkey
resultCallBack	StringResultCallBack	回调

响应示例:

```
{
    "code": 1,
    "data": {
        "status": 1
    },
    "msg": null
}
```

响应示例说明:

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
status	int	1 删除成功 0删除失败

十三、获取机器人引导语：robotGuide

如果在工作台设置了机器人引导语，那么可以调用此接口来获取机器人引导语。

robotGuide(String uid,String robotFlag,final StringResultCallBack<ZhiChiMessage> resultCallBack);

请求参数说明：

变量名	数据类型	说明
uid	String	用户id
robotFlag	String	机器人编号（可以为空）
resultCallBack	StringResultCallBack	回调

响应示例：


```
{
    "code": 1,
    "data": {
        "ustatus": -1,
        "answer": {
            "msgType": 0,
            "msg": "这是第二个机器人-1\n\n",
            "duration": null,
            "richpricurl": null,
            "richmoreurl": null
        },
        "answerType": "13",
        "stripe": null,
        "suggestionList": [
            {
                "question": "这是第二个机器人",
                "docId": "3e11e3b54d214b62ba16e2c26c69d869",
                "answer": "这是第二个机器人"
            }
        ],
    },
    "msg": null
}
```


响应示例说明：

字段名	数据类型	说明
code	String	请求结果码：成功code==1 失败 code==0
ustatus	int	
answer	String	富媒体消息 msgType 0文本 1图片 2音频 4 富文本中有图片 5 富文本中纯文字 6 富文本中有视频 msg消息内容 duration语音时长 richpricurl图片链接 richmoreurl点击更多链接
answerType	int	1 直接回答 2 理解回答 3 不能回答 4引导回答 6 互联网寒暄 7 私有寒暄（包括第三方天气、快递接口） 8 百科 9 向导回答 10 业务接口
stripe	String	引导建议
suggestionList	list	引导说辞带docId
suggestions	String[]	引导说辞不带docId
question	String	问题内容
docId	String	标题id
answer	String	答案

十四、用户主动下线接口：out

调用此接口后用户将在客服工作台中显示下线，同时将计入统计数据。调用此接口成功后，发送客服消息将无法成功。

out(String cid, String uid, StringResultCallBack <CommonModel> resultCallBack);

请求参数说明：

变量名	数据类型	说明
uid	String	用户id
cid	String	会话id
filePath	String	文件
duration	String	语音的时长
resultCallBack	StringResultCallBack	回调

响应示例：

```
{
    "code": 1,
    "data": {
        "status": 1
        }
}
```

响应示例说明：

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
status	int	1 发送成功 2发送失败

十五、留言上传图片接口： fileUploadForPostMsg

在提交留言时需要上传图片时可以使用此接口，上传完毕后将返回的文件路径以逗号分隔，作为参数 提交到留言接口（postMsg）中。

fileUploadForPostMsg(String companyId, String filePath,
ResultCallBack<ZhiChiMessage> resultCallBack);
请求参数说明:

字段名	数据类型	说明
companyId	String	公司ID
filePath	String	文件路径
resultCallBack	StringResultCallBack	回调

响应示例:


```
{
 "code": 1,
"data": {
        "url":"https://img.sobot.com/chatres/e6fb17f5af9f4487a8ec7f1f58c972ba/msg/20170605/dts_featured_ath_clr100_1496658840995.png"
        },
    "msg": null
}
```


响应示例说明:

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
url	String	文件在服务器上面的地址

十六、机器人回答顶踩接口： rbAnswerComment

此接口可以对机器人的回答词条进行命中率统计，调用后将计入统计数据。

rbAnswerComment(String uid, String cid, String robotFlag, String docId, String docName,boolean revaluateFlag, StringResultCallBack<CommonModelBase> resultCallBack);
请求参数说明:

字段名	数据类型	说明
uid	String	用户id
cid	String	会话Id
robotFlag	String	机器人ID
docId	String	词条id
docName	String	词条名称
revaluateFlag	boolean	反馈结果-顶/踩 true 顶 false 踩
resultCallBack	StringResultCallBack	回调

响应示例:


```
{
    "code": 1,
    "data": {
        "status": 1
    },
    "msg": null
}
```

响应示例说明：

字段名	数据类型	说明
code	int	请求结果码：成功code==1 失败 code==0
status	int	顶踩成功

十七、服务器推送的消息格式

需要注册广播：ZhiChiConstants.receiveMessageBrocast

1）用户与人工客服建立会话响应示例：

```
{
    "type": 200,
    "aname": "sobotcggghh",
    "aface": "https://img.sobot.com/chb0.jpg",
    "aid": "1b6d22ab52bc4db99c3c7b055abd3332",
}
```

响应参数说明：

字段名	数据类型	说明
type	int	状态
aname	String	客服名称
aface	String	客服头像
aid	String	客服id

2）用户排队响应示例：

```
{
    "type": 201,
    "count": 3,
}
```

响应参数说明：

字段名	数据类型	说明
type	int	状态
count	String	当前排队位置

3）用户收到客服新消息响应示例：

```
{
    "type": 202,
    "aname": "sobotgghh",
    "aface": "https://img.sobot.com/ch0424785f6443acbc350b0.jpg",
    "content": "",
    "msgType": *,
}
```

响应示例说明：

字段名	数据类型	说明
type	int	状态
aname	String	客服名称
aface	String	客服头像
content	String	客服发送的消息
msgType	Int	消息类型0文本 1图片 2音频 4 富文本中有图片

4）用户被下线响应示例：

```
{
    "type": 204,
    "status": 2,
    "aname": "sobothhh",
    "msgId": "6793ec14b4d44739a7dc289d7ecc26c7",
    "puid": "0953be7e19b3494e8c6da517e2d062c9"
}
```

响应示例说明：

字段名	数据类型	说明
type	int	状态
status	Int	下线原因 1、管理员下线2、被管理员移除3、被加入黑名单4、长时间未说话 6打开新窗口
aname	String	客服姓名

5）用户被客服转接响应示例：

```
{
    "type": 210,
    "name": "311714789@qq.com",
    "face": "https://img.sobot.com/chat8f5d475091ad7a400b04ad96.jpg",
}
```

响应示例说明：

字段名	数据类型	说明
type	int	状态
name	String	转接后的客服名称
face	String	转接后的客服头像

6）客服推送满意度评价响应示例：

```
{
    "type": 209,
    "aname": "311714789@qq.com",
    "aid": "442d6f93cce54134a13542dd66dea1e7",
    "msgId": "2326a269b9f04d6c8c318b8d7678c8a0"
}
```

响应示例说明：

字段名	数据类型	说明
type	int	状态209客服推送满意度评价
name	String	客服名称
aid	String	客服

代码混淆

```
-keepattributes Annotation
-keepattributes Signature
```

```
-keep public class * extends android.app.Fragment
-keep public class * extends android.app.Activity
-keep public class * extends android.app.Application
-keep public class * extends android.app.Service
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider
-keep public class * extends android.app.backup.BackupAgentHelper
-keep public class * extends android.preference.Preference
-keep public class * extends android.support.v4.**
-keep public class com.android.vending.licensing.ILicensingService
-keep class com.android.vending.licensing.ILicensingService
-keep class android.support.v4.** { *; }

-dontwarn android.support.v4.**
-dontwarn android.webkit.WebView
-keepclasseswithmembernames class * {
    native <methods>;
}

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet);
}

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet, int);
}

-keepclassmembers enum * {
    public static **[] values();
    public static ** valueOf(java.lang.String);
}

-keep class * implements android.os.Parcelable {
  public static final android.os.Parcelable$Creator *;
}

-keepclasseswithmembers class * {
    public <init>(android.content.Context);
}

-keepclassmembers class * {
   public <init> (org.json.JSONObject);
}
## ----------------------------------
##      sobot相关
```

```
## ----------------------------------
-keep class com.sobot.** {*;}
-keep class com.yunva.** { *; }
-keep class com.blm.** { *; }
-keep class com.pg.** { *; }
-dontwarn com.yunva.im.sdk.lib.**

## ----------------------------------
##      OkHttp相关
## ----------------------------------
-dontwarn okhttp3.**
-dontwarn okio.**
-dontwarn javax.annotation.**
-keepnames class okhttp3.internal.publicsuffix.PublicSuffixDatabase

## ----------------------------------
##      UIL相关
## ----------------------------------
-keep class com.nostra13.universalimageloader.** { *; }
-keepclassmembers class com.nostra13.universalimageloader.** {*;}
-dontwarn com.nostra13.universalimageloader.**

## ----------------------------------
##      Glide相关
## ----------------------------------
-keep class com.bumptech.glide.Glide { *; }
-keep class com.bumptech.glide.request.RequestOptions {*;}
-keep public class * implements com.bumptech.glide.module.GlideModule
-keep public enum
com.bumptech.glide.load.resource.bitmap.ImageHeaderParser$** {
  **[] $VALUES;
  public *;
}
-dontwarn com.bumptech.glide.**

## ----------------------------------
##      Picasso相关
## ----------------------------------
-keep class com.squareup.picasso.Picasso { *; }
-dontwarn com.squareup.okhttp.**
-dontwarn com.squareup.picasso.**
```

高度自定义UI

为了咨询客服窗口的界面风格能与集成智齿客服 SDK 的 App 整体统一，智齿客服 SDK 提

供了简洁的 UI 自定义配置选项，包括两种UI自定义方式：接口传入参数、修改资源文件。

智齿客服SDK允许对界面头部、聊天部分、底部栏进行UI自定义，比如图片，背景，文字大小以及颜色。

以下两种方式请结合实际情况使用。

一、接口传入参数

以下属性在application的oncreate()方法中设置

```
//设置 转人工按钮的图片
SobotUIConfig.sobot_serviceImgId = R.drawable.xxx;
//设置 头部文字字体颜色
SobotUIConfig.sobot_titleTextColor = R.color.xxx;
//设置 右上角按钮图片
SobotUIConfig.sobot_moreBtnImgId  = R.drawable.xxx;
//设置 头部背景颜色
SobotUIConfig.sobot_titleBgColor = R.color.xxx;
//设置 聊天界面底部整体布局背景颜色
SobotUIConfig.sobot_chat_bottom_bgColor = R.color.xxx;
//设置 聊天界面左边气泡内文字字体颜色
SobotUIConfig.sobot_chat_left_textColor = R.color.xxx;
//设置 聊天界面左边气泡内链接文字字体颜色
SobotUIConfig.sobot_chat_left_link_textColor = R.color.xxx;
//设置 聊天界面左边气泡背景颜色
SobotUIConfig.sobot_chat_left_bgColor = R.color.xxx;
//设置 聊天界面右边气泡背景颜色
SobotUIConfig.sobot_chat_right_bgColor = R.color.xxx;
//聊天界面文件消息类型气泡背景颜色
SobotUIConfig.sobot_chat_file_bgColor = R.color.xxx;
//设置 聊天界面右边气泡内链接文字字体颜色
SobotUIConfig.sobot_chat_right_link_textColor = R.color.xxx;
//设置 聊天界面右边气泡内文字字体颜色
SobotUIConfig.sobot_chat_right_textColor = R.color.xxx;


//设置 toolbar右边第二个按钮是否显示（评价）
SobotUIConfig.sobot_title_right_menu2_display = false;
//修改toolbar右边第二个按钮的图片
SobotUIConfig.sobot_title_right_menu2_bg = R.drawable.xxx;
```

二、修改资源文件

注意：本章文案的阅读对象是熟悉布局修改的Android开发人员。由于本lib库是UI库，每次升级都可能对UI做较大变动，因此如果本次自定义了一些图片等资源文件，建议不直接修改 lib库里的资源文件，而是找到对应资源文件或者对应资源的引用(colors.xml、strings.xml文件中的值)，将其修改后拷贝到主工程目录中，如果资源名称相同的情况下，开发工具会自 动使用主工程下的资源（例如需要修改一个颜色值，那么在sdk下的colos.xml找到对应的引用拷贝到您的colos.xml文件中即可），使用这种方式能极大的提高维护效率。

1 资源一览

布局文件（layout）

文件名	功能说明
sobot_chat_activity.xml	聊天界面
sobot_layout_titlebar.xml	聊天界面顶部导航栏
sobot_layout_chat_bottom.xml	聊天界面底部栏
sobot_upload_layout.xml	聊天界面底部选择图像功能
sobot_list_item_wo_text.xml	对话中客户文字类型布局
sobot_list_item_wo_image.xml	对话中客户图片类型布局
sobot_list_item_wo_voice.xml	对话中客户语音类型布局
sobot_list_item_robot_text.xml	对话中机器人(人工客服)文字类型布局
sobot_other_dialog.xml	结束对话窗口
sobot_dialog.xml	满意度评价窗口

配置文件

文件名	功能说明
colors.xml	颜色修改
dimens.xml	字体大小，图片显示大小
strings.xml	文案修改

资源文件夹

文件夹名	功能说明
drawable-xhdpi	存放UI布局使用的图片资源文件

2 资源定义详细说明

沉浸式状态栏

配置项	参数名	备注
状态栏颜色	colors.xml sobot_status_bar_color	

顶部导航栏

layout: sobot_layout_titlebar.xml

配置项	参数名	备注
背景颜色	colors.xml sobot_color_title_bar_bg	
中间文字	TextView: sobot_text_title	
中间文字大小	dimens.xml sobot_text_title	
中间文字颜色	colors.xml sobot_color_title_bar_title	
返回按钮	TextView: sobot_tv_left	
返回按钮文字	sobot_back	
返回按钮文字颜色	sobot_bg_white	
返回按钮文字大小	dimens.xml sobot_text_title	
返回按钮资源文件	sobot_icon_back_pressed .png sobot_icon_back_normal. png	

聊天区域背景

layout: sobot_chat_activity.xml

配置项	参数名	备注
聊天区域背景颜色	colors.xml sobot_lv_message_bg	DropdownListView: sobot_lv_message

提示气泡（时间、接通客服等提示）

时间、接通客服提示：sobot_toast_selector.xml

配置项	参数名	备注
提示气泡	TextView: sobot_center_Remind_note	
时间提示气泡样式	style:sobot_center_remind_time	
消息提示气泡样式	style:sobot_center_remind_note	
提示气泡文字颜色	colors: sobot_listview_remind_text_color	
超链颜色	colors:sobot_color_link_remind	
提示气泡文字大小	dimens:sobot_listview_remind_text_size	

客服消息（文字类型、图片类型）

文字类型 layout: sobot_chat_msg_item_txt_l.xml

图片类型 layout: sobot_chat_msg_item_imgt_l.xml

配置项	参数名	备注
气泡背景	drawable-xhdpi sobot_chatfrom_bg_normal.9.png	LinearLayout: sobot_ll_content
文字颜色	colors.xml robot_msg_text_color	TextView: sobot_msg
超链颜色	colors.xml sobot_color_link	
文字大小	dimens.xml robot_msg_text_size	
阅读全文	colors.xml sobot_color_read_all	TextView:sobot_rendAllText

用户消息（文字类型、图片类型、语音类型）

文字类型 layout: sobot_chat_msg_item_txt_r.xml

图片类型 layout:sobot_chat_msg_item_imgt_r.xml

语音类型 layout: sobot_chat_msg_item_audiot_r.xml

配置项	参数名	备注
气泡背景	drawable-xhdpi sobot_chatto_bg_normal.9.png	LinearLayout: sobot_ll_content
文字颜色	colors.xml sobot_msg_text_color	TextView: sobot_msg
超链颜色	colors.xml sobot_color_rlink	
文字大小	dimens.xml sobot_msg_text_size	
语音时长		TextView: sobot_voiceTimeLong
语音时长字体颜色	colors.xml sobot_msg_voice_text_color	
语音时长字体大小	dimens.xml sobot_msg_voice_text_size	
播放语音帧动画		ImageView: sobot_iv_voice
播放语音默认背景	drawable-xhdpi sobot_chatto_bg_normal.9.png	
帧动画的资源文件	drawable-xhdpi sobot_pop_voice_send_anime_1.png(以sobot_pop_voice_send_anime_开头多张)	drawable:sobot_voice_to_icon.xml

满意度评价窗口

layout:sobot_layout_evaluate.xml

配置项	参数名	备注
提交评价按钮背景	drawable sobot_text_button_selector	Button
提交评价按钮文字颜色	drawable sobot_text_button_color_selector	Button
评价选择背景色	drawable sobot_evaluate_btn_selector	
评价选择文字颜色	drawable sobot_btn_text_color_selector	

输入栏和加号菜单

底部 layout: sobotlayoutchat_bottom.xml

图片选择 layout: sobot_upload_layout.xml

配置项	参数名	备注
背景	colors.xml sobot_color_bottom_bg	LinearLayout: sobot_rl_bottom
人工转接按钮	drawable-xhdpi sobot_icon_manualwork_pressed.png，sobot_icon_manualwork_normal.png	Button: sobot_btn_set_mode_rengong
键盘按钮	drawable-xhdpi sobot_keyboard_pressed.png, sobot_keyboard_normal.png	Button: sobot_btn_model_edit
语音按钮	drawable-xhdpi sobot_icon_vioce_pressed.png，sobot_icon_vioce_normal.png- xhdpi	Button: sobot_btn_model_voice
上传图片	drawable-xhdpi sobot_picture_pressed.png, soot_picture_normal.png	Button: sobot_btn_upload_view
发送	drawble sobot_btn_sendmsg_selector.xml	Button: sobot_btn_send
输入框背景色	colors.xml sobot_color_bottom_edittext_text	ContainsEmojiEditText:sobot_et_sendmessage
按住说话按钮父背景	drawble sobot_chat_press_speak_btn.xml	LinearLayout:sobot_btn_press_to_speak
按住说话按钮文字颜色	colors.xml sobot_color_bottom_btn_voice	TextView:sobottxtspeak_content
按住说话按钮文字大小	dimens .xml sobot_text_font_normal	TextView:sobot_txt_speak_content

图片选择

图片选择 layout: sobot_upload_layout.xml

背景 drawable: sobot_linearlayout_border.xml

配置项	参数名	备注
相册按钮背景	drawable sobot_login_edit_nomal1.xml	TextView:sobot_btn_picture
相册按钮中小图	drawable-xhdpi sobot_uploadpicture.png	
相册按钮文字颜色	color.xml sobot_color_bottom_upload_btn_color	
相册按钮文字大小	dimens .xml sobot_text_font_small	
拍照	TextView:sobot_btn_take_picture drawable sobot_login_edit_nomal1.xml	
拍照按钮中小图	drawable-xhdpi sobot_takephoto.png	
拍照按钮文字颜色	color.xml sobot_color_bottom_upload_btn_color	
拍照按钮文字大小	dimens .xml sobot_text_font_small	

开始新会话(LinearLayout:sobot_ll_restart_talk)

layout: sobot_layout_chat_bottom.xml

配置项	参数名	备注
背景	drawable sobot_chat_press_speak_btn1.xml	
文字颜色	color.xml sobot_txt_restart_talk	

修改初始化Loading页logo
布局：sobot_layout_chat_loading.xml
将准备好的资源文件命名为sobot_loding.gif。
将此资源文件放到项目的drawable-hdpi文件夹下面

附：智齿 Android_SDK 源码 Git下载地址：https://github.com/ZCSDK/SobotSDK_Android
常见问题j解答：https://github.com/ZCSDK/FAQ_Android

联系我们