AnyChat SDK for Android

开发流程指南

(版本：V2.3)

广 州 佰 锐 网 络 科 技 有 限 公 司

GuangZhouBaiRui Network Technology Co.,Ltd.

http://www.bairuitech.com http://www.anychat.cn

2015 年 07 月

目录

一、简介

AnyChat SDK(AnyChat 音视频互动开发平台)是一套跨平台的音视频即时通讯解决方案，基于先进的 H.264 视频编码标准、AAC 音频编码标准与 P2P 技术，支持高清视频，整合了佰锐科技在音视频编码、多媒体通讯领域领先的开发技术和丰富的产品经验而设计的高质量、宽适应性、分布式、模块化的网络音视频互动平台。

基于 Android 的客户端 SDK 应用于 Android2.3 以上版本的设备，您可以通过该套 SDK API 接口实现在 Android 平台快速开发基于音视频通讯交互功能的 App 程序，主要提供的功能如下：

- 音视频即时通讯：提供语音、视频一对一、一对多的实时通讯，支持高清视频和高品质音频效果。
- 录像：支持针对单个人的音视频录制、整个视频通话过程内容的合成音视频录制以及集中服务器保存录制
- 抓拍：可对本地视频和正在视频的对象进行抓拍；
- 文字聊天：支持多用户之间的文字交流；
- 透明通道：提供客户端之间、客户端跟服务器之间的数据通讯能力；
- 文件传输：支持客户端直接、客户端跟服务器之间的文件传输功能，支持断点续传；
- 动态设置音视频参数：提供音视频参数设置的接口，可以根据需要动态设置分辨率、码率、帧率等视频参数，满足各种应用场景的需求；
- 外部音视频输入：支持非标准采集设备以外的音视频源输入，满足更多的应用场景；
- 集成第三方外部音视频编解码器：可集成第三方音视频编解码器，满足特殊环境下面的硬件编解码要求；
- 业务排队：提供自定义营业区域、队列功能，实现客户排队、坐席为队列中客户提供服务的功能；

1.1 面向的读者

《AnyChat SDK for Android开发流程指南》文档是提供给具有一定的Android编程经验和了解面向对象概念的读者使用，不要求具备音视频开发方面的经验。您在使用遇到任何问题，都可以通过访问bbs.anychat.cn反馈给我们。

1.2 获取 AnyChat SDK for Android

您可在AnyChat的产品官方网站下载到最新版的AnyChat for Android SDK，下载地址为: http://www.anychat.cn/download.html，如下图所示：

AnyChat for Android SDK 包里面提供了 demo 程序的编译程序、开发指南、demo 程序源码和 SDK 文件，其解压之后的目录结构如下所示：

|----bin AnyChat SDK 演示程序（安装包）

|----doc 客户端开发指南

|----src Demo 程序源代码（基于 android2.3.3 工程）

|----sdk 客户端 SDK 引用文件

1.3 技术支持

在您使用本 SDK 的过程中，遇到任何困难，请与我们联系，我们将热忱为您提供帮助。

您可以通过如下方式与我们取得联系：

1、 在线论坛：http://bbs.anychat.cn/

2、 知识中心：http://www.anychat.cn/faq/

3、 官方网站：http://www.anychat.cn

4、 电子邮件：service@bairuitech.com

5、 24 小时客服电话：+86 （020） 85276986、38109065、38103410

二、编写说明

本指南的编写是为了帮助使用 AnyChat for Android SDK 的用户快速地搭建 SDK 开发环境、熟悉 SDK 开发流程、掌握 SDK 开发功能接口而编写的，

其中“工程准备”、“基本流程”、“音视频交互”三章的内容是基于 src 目录中提供的 HelloAnyChat 工程来编写说明的，涉及到的开发环境配置、以及相关代码说明可以参考 HelloAnyChat 工程源码。

其中“业务排队”、“呼叫中心”章节内容是基于 src 目录中提供的 AnyChatQueue 工程来编写说明的，涉及到的开发环境配置、以及相关代码说明可以参考 AnyChatQueue 工程源码。

三、工程准备

Android 开发工具有很多，开发者可根据自己的喜好进行选择。在此，我们推荐开发者使用 eclipse 作为自己的开发工具，本套开发指南也是针对 Eclipse 开发环境下进行编写的。在 eclipse 中新建一个 Android 工程，对工程进行以下配置，搭建 AnyChat 的开发环境。

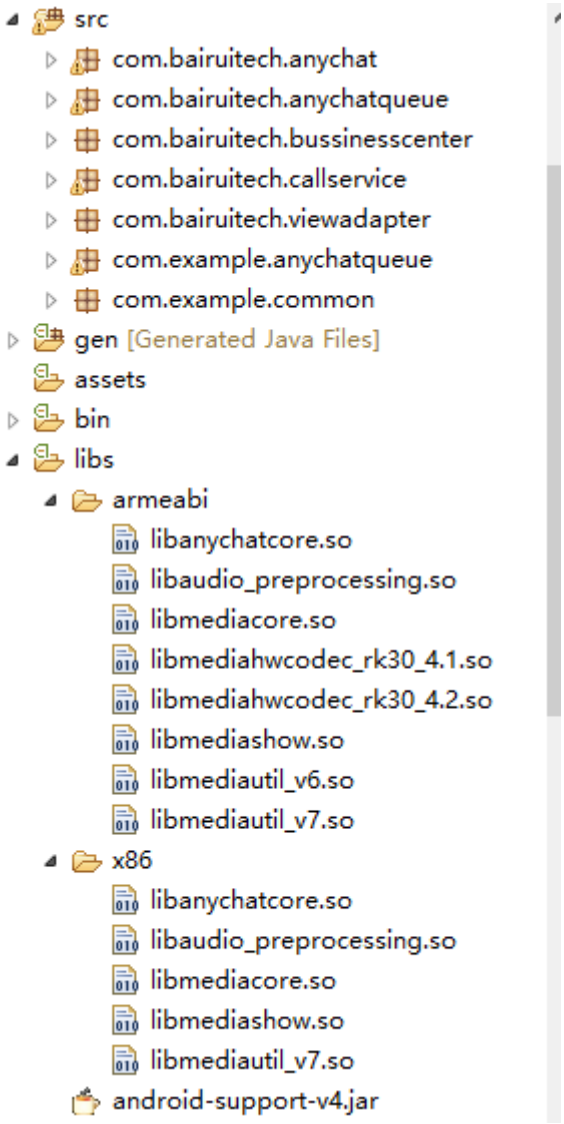
3.1 导入库文件

本 SDK 支持 armeabi 和 x86 两大架构的 android 设备，并为其提供了相应的库文件，在 AnyChatQueue 工程的 libs 目录下，有子目录 armeabi（适用于 ARM 架构）、子目录 x86（适用于 x86 架构）分别对应两种架构，建议在 apk 打包时将这两个子目录同时打包，这样同一个 apk，便可以兼容 ARM 和 x86 的设备。

如果想缩小 apk 包的体积，只支持 ARM 架构的 Android 设备，则可以删除 x86 目录；

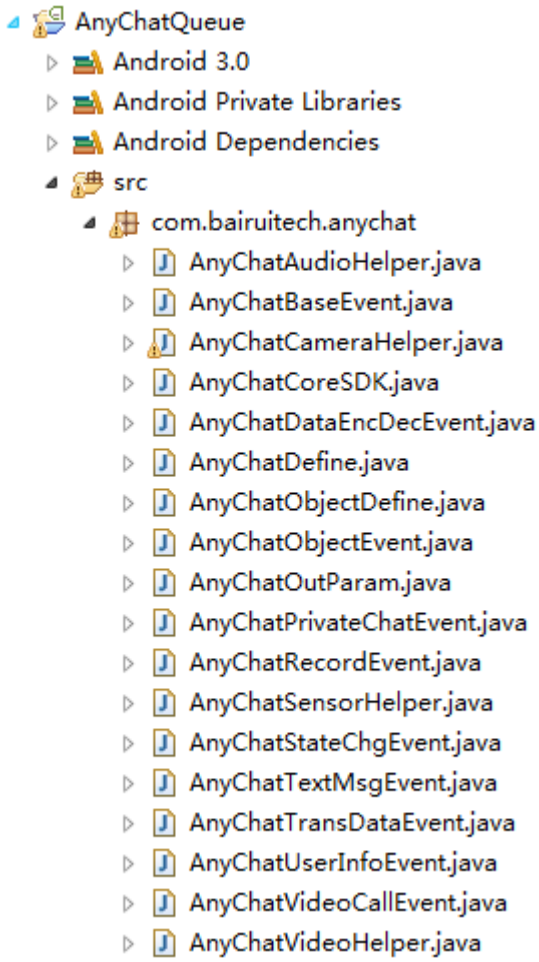
如果只需要支持 x86 架构的 Android 设备，则可以删除 armeabi 目录（如果您工程中有建立 libs\armeabi-v7a 的话，也需要拷贝一份 armeabi 目录下的库文件到这个目录）。

将 AnyChatQueue 工程中 lib 下的库文件拷贝到你的工程中，拷贝完成后的工程目录如下图所示：



3.2 导入 SDK 文件

在工程的 src 目录下新建 com.bairuitech.anychat 包（注意：包名一定要是这个，不能更改），将开发包中的 sdk 目录下面的所有 Java 文件拷贝到新建的包中，拷贝后的工程目录如下图所示：



3.3 添加权限

添加相关权限使得 SDK 包接口可以正常使用，需要添加的权限如下：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.RESTART_PACKAGES" />
```

四、基本流程

在工程准备好了之后，需要实现以下基本流程，才能调用音视频交互等其他功能接口。

4.1 准备一个 AnyChatCoreSDK 对象

AnyChatCoreSDK 类是 SDK 的核心类，提供各种功能接口，如连接服务器、登录、进入房间、操作音视频等。在使用这些功能接口构建应用之前，需要准备一个 AnyChatCoreSDK 对象，这个对象可以是单例模式，在其他 activity 中共享，也可以在每个 activity 中都新建一个。

方法一：新建一个 AnyChatCoreSDK 对象，参考代码如下：

```
//新建核心类对象
 AnyChatCoreSDK anychat=null;
 anychat = new AnyChatCoreSDK();
```

方法二：从 AnyChatCoreSDK 类中获取一个单例对象，参考代码如下：

```
//获取核心类对象
 AnyChatCoreSDK anychat=null;
 anychat=AnyChatCoreSDK.getAnyChatInstance(null);
```

4.2 监听基本事件和业务对象事件

监听“连接服务器、用户登录、进入房间、与服务器网络连接”等事件。在需要接收的 activity 或者类中实现以下两步。

(1) 继承并实现 AnyChatBaseEvent 和 AnyChatObjectEvent 接口，参考代码如下：

```
public class HelloWorldAnyChat extends Activity implements AnyChatBaseEvent
，AnyChatObjectEvent{
//连接服务器触发（connet），“bSuccess==true”连接服务器成功，反之连接服务器失　败
@Override
public void OnAnyChatConnectMessage(boolean bSuccess)
{
}
//用户登录触发(login),dwUserId是服务器为客户端分配的唯一标识userid，dwErrorCode==0
表示登录成功，其他值为登录服务器失败的错误代码
@Override
public void OnAnyChatLoginMessage(int dwUserId, int dwErrorCode)
{
}
//进入房间触发，dwRoomId为房间号，dwErrorCode==0表示进入房间成功，其他值为进入 房　间
失败的错误代码
@Override
public void OnAnyChatEnterRoomMessage(int dwRoomId, int dwErrorCode)
 {
 }
// 当前房间在线用户消息，进入房间成功后调用一次。dwUserNum当前房间总人数（包含自　己）
@Override
public void OnAnyChatOnlineUserMessage(int dwUserNum, int dwRoomId)
 {
 }
// 当前房间用户离开或者进入房间触发这个回调，dwUserId用户 id," bEnter==true"表示进入
房间,反之表示离开房间
@Override
public void OnAnyChatUserAtRoomMessage(int dwUserId, boolean bEnter)
{
}
// 跟服务器网络断触发该消息。收到该消息后可以关闭音视频以及做相关提示工作
@Override
public void OnAnyChatLinkCloseMessage(int dwErrorCode)
{
}
//业务对象回调事件，调用AnyChatCoreSDk.ObjectControl方法触发这个回调
@Override
public void OnAnyChatObjectEvent(int dwObjectType, int dwObjectId,
            int dwEventType, int dwParam1, int dwParam2, int dwParam3,
            int dwParam4, String strParam) {}
```

(2) 设置基本回调事件接收，参考代码如下：

```
// 设置基本回调事件接收
anychatSDK.SetBaseEvent(this);
// 设置业务对象回调事件接收
anychatSDK.SetObjectEvent(this);
```

4.3 初始化 SDK

加载资源，应用程序中只需要执行一次，其他的功能接口都必须在初始化之后才能正常使用，参考代码如下：

```
// 初始化SDK，第一个参数参入的是android的系统版本号
anychat.InitSDK(android.os.Build.VERSION.SDK_INT, 0);
```

4.4 连接、登录服务器

使用AnyChat功能的通讯功能，必须要先连接登录AnyChat的通讯服务器（AnyChat通讯服务器的部署参考链接：http://bbs.anychat.cn/forum.php?mod=viewthread&tid=8&extra=page%3D1），这里以我们对外公开测试服务器地址demo.anychat.cn为例，参考代码如下：

```
// 连接服务器,第一个参数为你需要连接的视频服务器地址，如果您部署视频服务器的地址为192.168.1.8，则传入这个地址
anychat.Connect("www.anychat.cn", 8906);
//用户登录
anychat.Login("yourname", "");
```

这两个流程的执行都是异步的操作，会触发相应的 OnAnyChatConnectMessage、OnAnyChatLoginMessage 函数，可以连续调用，也可以在上一步的回调中（参考 3.2）执行下一步操作，参考代码如下：

```
// 连接服务器
anychat.Connect("demo.anychat.cn", 8906);
@Override
public void OnAnyChatConnectMessage(boolean bSuccess) {
//登录服务器
anychat.Login("AnyChat", "");
}
public void OnAnyChatLoginMessage(int dwUserId, int dwErrorCode)
{
}
```

4.5 进入房间

除了音视频交互功能需要本流程之外，没有特殊说明，其他功能都不需要本流程。应用层将 roomid 传入，进入指定的房间，只有在同一个房间内的用户才能进行音视频交互，参考代码如下：

```
//进入房间
AnyChatCoreSDK.EnterRoom(1,"",0);
```

此流程操作是一个异步的操作，会依次触发消息回调。

```
//用户进入房间的事件
    public void OnAnyChatEnterRoomMessage(int dwRoomId, int dwErrorCode) {

    }
    //在线用户信息
    public void OnAnyChatOnlineUserMessage(int dwUserNum, int dwRoomId) {

    }
```

后续的音视频互动见音视频模块详情。

五、音视频交互

AnyChat for Android SDK 为开发者提供了便捷的建立音视频通讯的接口，通过以下几步操作，即可在您的应用中集成音视频交互功能。需要注意的是只有在同一个房间内的用户才能进行音视频通讯。

5.1 准备 surfaceview 控件

(1) 在布局文件中添加 surfaceview 控件，用于显示视频。sufaceview 的大小可以自定义，决定显示视频的大小。参考代码如下：

```
<!--本地视频 surfaceview-->
<SurfaceView
   android:id="@+id/surfaceview_local"
    android:layout_width="320dip"
    android:layout_height="240dip" />
<!--远程视频 surfaceview-->
<SurfaceView
    android:id="@+id/surfaceview_remote"
    android:layout_width="320dip"
    android:layout_height="240dip" />
```

(2) 在 activity 的 oncreate 中实例化 surfaceview 对象。

```
// 初始化用于显示视频的控件 sufaceview
mSurfaceLocal = (SurfaceView) this.findViewById(R.id.surfaceview_local);
mSurfaceRemote = (SurfaceView) this.findViewById(R.id.surfaceview_remote);
```

5.2 设置必要的参数

在初始化之后，需要设置必要的音视频参数，参考代码如下：

```
//设置本地视频采集随着设备而旋转而处理
AnyChatCoreSDK.SetSDKOptionInt(AnyChatDefine.BRAC_SO_LOCALVIDEO_AUTOROTATI
ON, 1);
```

5.3 摄像头硬件初始化

该操作需要在显示视频 activity 的 oncreate 中执行（**注意：必须是 OnCreate 方法中初始化 Camera 设备**），参考代码如下：

```
// 启动 AnyChat 传感器监听
anychat.mSensorHelper.InitSensor(this);
// 初始化 Camera 上下文句柄
AnyChatCoreSDK.mCameraHelper.SetContext(this);
//设置 SURFACE_TYPE_PUSH_BUFFERS 模式
mSurfaceLocal.getHolder().setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
// 打开本地视频预览，开始采集本地视频数据
mSurfaceLocal.getHolder().addCallback(AnyChatCoreSDK.mCameraHelper);
```

5.4 打开本地音视频

调用 UserCameraControl 打开视频，调用 UserSpeakControl 打开音频。打开本地音视频数据需要在进入房间成功之后才有效，即在收到 OnAnyChatEnterRoom 回调后（参考 4.5）打开本地音视频，其他客户端才能请求到你的音视频数据。参考代码如下：

```
//打开本地视频，第一个参数用-1 表示本地，也可以用本地的真实 userid
anychat.UserCameraControl(-1, 1);
//打开本地音频
anychat.UserSpeakControl(-1, 1);
```

5.5 关闭本地音视频

打开本地音视频后，可以在音视频交互的过程中选择关闭本地音视频。同时，还可以在关闭之后重新打开本地音视频（参考 5.4；在音视频交互结束之后需要调用该操作，释放本地摄像头和音频采集设备，参考代码如下：

```
//关闭本地视频，第一个参数用-1 表示本地，也可以用本地的真实 userid
anychat.UserCameraControl(-1, 1);
//关闭本地音频
anychat.UserSpeakControl(-1, 1);
```

5.6 请求远程音视频

请求通话目标用户的音视频数据，调用 UserCameraControl 打开视频，调用 UserSpeakControl 打开音频，bindVideo、SetVideoUser 绑定显示视频的 surfaceview（mSurfaceRemote 为指定显示远程视频的 surfaceview；如要显示多个人的视频，准备多个 surfaceview，为每个用户绑定一个即可），参考代码如下：

```
// mRemoteUserid 为通话目标对象的 userId;
int index = anychat.mVideoHelper.bindVideo(mSurfaceRemote.getHolder());
anychat.mVideoHelper.SetVideoUser(index, mRemoteUserid);
anychat.UserCameraControl(mRemoteUserid, 1);
anychat.UserSpeakControl(mRemoteUserid, 1);
```

在触发 OnAnyChatOnlineUser 或者 OnAnyChatEnterRoom 并判断通话目标对象已经进入当前房间之后（参考 4.2），该操作才有效。

5.7 关闭远程音视频

请求远程音视频后，可以在音视频交互的过程中选择关闭远程音视频。同时，还可以在关闭之后重新请求远程音视频（参考 5.6）；在音视频交互结束之后需要调用该操作，释放远程音视频资源，参考代码如下：

```
//关闭远程视频, mRemoteUserid 为通话目标的 userid
anychat.UserCameraControl(mRemoteUserid, 0);
//关闭远程音频,
anychat.UserSpeakControl(mRemoteUserid, 0);
```

六、业务排队

AnyChat for Android SDK 为开发者提供了实现业务队列及用户排队、座席为队列内的用户进行服务的接口，通过以下几步操作，即可在您的应用中集成业务排队功能。座席从队列中取出用户后，通过调用视频呼叫的接口实现音视频交互。

AnyChat排队业务解决方案的介绍可以点击 “AnyChat提供业务排队整体解决方案” 链接查看。

6.1 初始化服务对象

```
public void OnAnyChatLoginMessage(int dwUserId, int dwErrorCode)
{
    If(dwErrorCode == 0){
//业务对象身份初始化：0代表普通客户，2是代表座席，3是代表经理；(dwUserflags)
    AnyChatCoreSDK.SetSDKOptionInt(AnyChatDefine.BRAC_SO_OBJECT_INITFLAGS,
dwUserFlags);
//业务对象优先级设定：默认为10
int dwPriority = 10;
AnyChatCoreSDK.ObjectSetIntValue(AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_CLIE
NTUSER,dwUserId,AnyChatObjectDefine.ANYCHAT_OBJECT_INFO_PRIORITY, dwPriority);
//业务对象属性设定,默认为 -1：
int dwAttribute = -1;
AnyChatCoreSDK.ObjectSetIntValue(AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_CLIE
NTUSER,      dwUserId,      AnyChatObjectDefine.ANYCHAT_OBJECT_INFO_ATTRIBUTE,
dwAttribute);
```

6.2 显示/进入/离开营业厅

在业务对象身份初始化之后，需要调用 ObjectControl 接口发送服务区域数据同步请求指令同步已存在的营业厅数据，接口第二个参数为-1 表示同步所有的营业厅。参考代码如下：

```
// 向服务器发送数据同步请求指令，此时会触发OnAnyChatObjectEvent的数据更新(UPDATE)回调
AnyChatCoreSDK.ObjectControl(AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_AREA,
AnyChatObjectDefine.ANYCHAT_INVALID_OBJECT_ID,
AnyChatObjectDefine.ANYCHAT_OBJECT_CTRL_SYNCDATA, mSelfUserId, 0, 0, 0, "");}
}
```

调用同步数据方法之后会接收到 ANYCHAT_OBJECT_EVENT_UPDATE 事件（业务对象更新事件），需要响应此事件去处理营业厅显示的逻辑功能，每个营业厅对象都会触发一次。参考代码如下：

```
 switch (dwEventType) {

// 1.业务对象数据更新事件，进入营业厅大厅和进入某个营业厅触发

case AnyChatObjectDefine.ANYCHAT_OBJECT_EVENT_UPDATE:              break;

switch (dwObjectType)
                {  //进入营业厅触发
                    case AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AREA:
                   //获取营业厅名称
                   list< String > list = new ArrayList< String >;
                   String name =
AnyChatCoreSDK.ObjectGetStringValue(AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_ARE
A,mobject.get(i), AnyChatObjectDefine.ANYCHAT_OBJECT_INFO_NAME);
                   list.add(name);
                   break;
                   //进入队列触发，获取名称方法和营业厅相同
                    case AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AREA:
                   break;
              }
```

将营业厅显示在界面之后，可以调用 ObjectControl 接口，执行 ANYCHAT_AREA_CTRL_USERENTER 方法进入某个营业厅。参考代码如下：

```
//进入营业厅
                    int retCode =
AnyChatCoreSDK.ObjectControl(AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AREA, tAreaID,
AnyChatCoreSDK.ANYCHAT_AREA_CTRL_USERENTER, 0, 0, 0, 0, string.Empty);
```

在执行成功后，会触发 ANYCHAT_AREA_EVENT_ENTERRESULT 事件（用户进入营业厅事件）。

如果要退出某个营业厅，也可以调用 BRAC_ObjectControl 接口，执行 ANY CHAT_AREA_CTRL_USER LEAVE 方法进入某个营业厅。参考代码如下：

```
//退出营业厅 ,iAreaID是当前营业厅的Id
AnyChatCoreSDK.ObjectControl(AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AREA, iAreaID,
AnyChatCoreSDK.ANYCHAT_AREA_CTRL_USERLEAVE, 0, 0, 0, 0, string.Empty);
```

6.3 显示/进入/离开队列

身份为“客户”的用户在进入营业厅后，在 ANYCHAT_AREA_EVENT_ENTERRESULT 事件（用户进入营业厅事件）中可以实现显示营业厅中的队列功能，会调用到 ObjectGetIdList、ObjectGetValue 等接口获取队列的信息，如队列 ID 列表、队列名称、队列描述信息、队列中当前人数等。参考代码如下：

```
switch (dwEventType) {
   // 1.业务对象数据更新事件，进入营业厅大厅和进入某个营业厅触发
   case AnyChatObjectDefine.ANYCHAT_AREA_EVENT_ENTERRESULT:
       //获取队列的Id数组
       int[]  queueIds  =
       AnyChatCoreSDK.ObjectGetIdList(AnyChatObjectDefine.ANYCHAT_OBJECT_T
       YPE_QUEUE);
       //通过QueueId查找名称
       String name = AnyChatCoreSDK.ObjectGetStringValue(
       AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_QUEUE,queueIds[i],
       AnyChatObjectDefine.ANYCHAT_OBJECT_INFO_NAME);
   break;
  default:   break;
}
```

在已显示的队列中，通过调用 ObjectControl 接口，执行 ANYCHAT_QUEUE_CTRL_USERENTER 方法进入某个队列。参考代码如下：

```
AnyChatCoreSDK.ObjectControl(AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_QUEUE
,Id, AnyChatObjectDefine.ANYCHAT_QUEUE_CTRL_USERENTER, 0, 0, 0, 0, "");
```

在进入队列后就开始排队，等待坐席的服务，在排队期间可以通过 BRAC_ObjectGetValue 接口获取队列的属性信息，如：获取当前队列人数、获取排在自己前面的用户数、自己在队列中的等待时间等。参考代码如下：

```
//获取队列等待时间，单位为秒
Int seconds = AnyChatCoreSDK.ObjectGetIntValue(
            AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_QUEUE,configEntity.Curre
            ntQueueId,
            AnyChatObjectDefine.ANYCHAT_QUEUE_INFO_WAITTIMESECOND);
//获取队列排队人数
Int number = AnyChatCoreSDK.ObjectGetStringValue(
AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_QUEUE,queueIds[i],
AnyChatObjectDefine. ANYCHAT_QUEUE_INFO_LENGTH);
//获取队列自己前面的用户数
Int number = AnyChatCoreSDK.ObjectGetStringValue(
AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_QUEUE,queueIds[i],
AnyChatObjectDefine. ANYCHAT_QUEUE_INFO_BEFOREUSERNUM);
```

在等待的过程中，通过调用 ObjectControl 接口，执行 ANYCHAT_QUEUE_CTRL_USERLEAVE 方法实现客户退出队列功能。参考代码如下：

```
AnyChatCoreSDK.ObjectControl(  AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_QUEUE,
dwObjectId,AnyChatObjectDefine. ANYCHAT_QUEUE_EVENT_LEAVERESULT, 0, 0, 0,0, "");
```

6.4 坐席服务

身份为“坐席”的用户在进入营业厅后，在 ANYCHAT_AREA_EVENT_ENTERRESULT 事件（用户进入营业厅事件）中可以进入到坐席服务窗口，会调用到 ObjectGetValue 等接口获取营业厅、队列等信息，如队列数量、队列总用户数、累计服务的用户数等。参考代码如下：

```
 //队列数
 int queueCount       =     AnyChatCoreSDK.ObjectGeIntValue(
AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AREA, AreaID,
AnyChatCoreSDK.ANYCHAT_AREA_INFO_QUEUECOUNT);

//队列总用户数
int queuesUserCount          =          AnyChatCoreSDK.ObjectGetIntValue(
AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AREA, AreaID,
AnyChatCoreSDK.ANYCHAT_AREA_INFO_QUEUEUSERCOUNT);

//累计服务的用户数
int servicedUserCount     =      AnyChatCoreSDK.ObjectGetIntValue(
AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AGENT, m_UserId,
AnyChatCoreSDK.ANYCHAT_AGENT_INFO_SERVICETOTALNUM);
```

坐席端通过调用 ObjectControl 接口，执行 ANYCHAT_AGENT_CTRL_SERVICEREQUEST 方法开始服务，从队列中获取一个用户（根据优先级、优先队列、先进先出等策略由系统自动分配）进行服务。

```
//客服开始服务
AnyChatCoreSDK.ObjectControl(AnyChatCoreSDK.ANYCHAT_OBJECT_TYPE_AGENT, m_UserId,
AnyChatCoreSDK.ANYCHAT_AGENT_CTRL_SERVICEREQUEST, 0, 0, 0, 0, "");
```

成功后会触发 ANYCHAT_AGENT_EVENT_SERVICENOTIFY 事件（坐席服务通知（哪个用户到哪个坐席办理业务）事件），同时被选择的用户会离开队列。在此事件中将进行坐席与客户之间的视频呼叫请求处理，参考代码如下：

```
public void OnAnyChatObjectEvent(int dwObjectType, int dwObjectId,
            int dwEventType, int dwParam1, int dwParam2, int dwParam3,
            int dwParam4, String strParam)

        switch (dwEventType) {
        case AnyChatObjectDefine.ANYCHAT_AGENT_EVENT_SERVICENOTIFY:
            //当客户角色是座席（服务器发的一个广播所有的座席都会触发到故需加限制条件)
            if(mApplication.getUserType() == 2 && dwParam1 ==
BussinessCenter.selfUserId)
            {
            anychat.VideoCallControl(AnyChatDefine.BRAC_VIDEOCALL_EVENT_REQUEST,
dwParam2, 0, 0, 0, "");
            }
            break;
```

后续的坐席与客户之间视频呼叫功能见下面“第七、视频呼叫”章节描述。

七、视频呼叫

AnyChat for Android SDK 为开发者提供了视频呼叫的功能，实现两个用户之间（如客户与坐席）的视频呼叫请求（Request）、视频呼叫回复（Reply）、视频呼叫开始（Start）以及视频呼叫结束（Finish）等过程，可以形象理解为打电话的流程：拨号、等待、通话、挂断。

AnyChat视频呼叫详细技术实现可以点击“AnyChat视频呼叫业务逻辑详解”链接查看。

7.1 视频呼叫请求

视频呼叫请求由请求方（如坐席）向被服务的一方（如客户）发起，通过调用 VideoCallEvent 接口（视频呼叫控制接口）去发送 VIDEOCALL_EVENT_REQUEST 事件实现发送视频呼叫请求，参考代码如下：

```
public void OnAnyChatObjectEvent(int dwObjectType, int dwObjectId,
            int dwEventType, int dwParam1, int dwParam2, int dwParam3,
            int dwParam4, String strParam)

        switch (dwEventType) {
        case AnyChatObjectDefine.ANYCHAT_AGENT_EVENT_SERVICENOTIFY:
            //当客户角色是座席（服务器发的一个广播所有的座席都会触发到故需加限制条件)
            if(mApplication.getUserType() == 2 && dwParam1 ==
BussinessCenter.selfUserId)
            {

    anychat.VideoCallControl(AnyChatDefine.BRAC_VIDEOCALL_EVENT_REQUEST,
dwParam2, 0, 0, 0, "");
            }
            break;
```

被服务的一方（如客户）接收到 VIDEOCALL_EVENT_REQUEST 事件后要实现是否接收请求的业务逻辑处理，参考代码如下：

```
            BussinessCenter.getBussinessCenter().onVideoCallRequest(
                    dwUserId, dwFlags, dwParam, userStr);
            if (dialog != null && dialog.isShowing())
                dialog.dismiss();
            dialog = DialogFactory.getDialog(DialogFactory.DIALOGID_REQUEST,
                    dwUserId, this);
            dialog.show();
            break;
```

7.2 视频呼叫回复

视频呼叫回复由被服务的一方（如客户）发起，通过调用 VideoCallControl 接口（视频呼叫控制接口）去发送 VIDEOCALL_EVENT_REPLY 事件实现发送视频呼叫回复；另外被服务的一方（如客户）也可以选择拒绝请求方（如坐席）的视频呼叫请求，参考代码如下：

```
//发送视频呼叫回复指令，dwErrcode=0
AnyChatCoreSDK.ObjectControl(AnyChatObjectDefine.ANYCHAT_OBJECT_TYPE_AGENT,
BussinessCenter.selfUserId,
AnyChatObjectDefine.ANYCHAT_AGENT_CTRL_SERVICEREQUEST, 0, 0, 0, 0, "");
```

传入 *BRAC_ERRORCODE_SESSION_REFUSE* 参数实现拒绝呼叫请求，参考代码如下：

```
//发送视频呼叫回复指令，拒绝请求，dwErrcode=100104
BussinessCenter.VideoCallControl(
                        AnyChatDefine.BRAC_VIDEOCALL_EVENT_REPLY, userId,
                        AnyChatDefine.BRAC_ERRORCODE_SESSION_REFUSE, 0, 0,
                        "");;
```

请求方（如坐席）在收到 BRAC_VIDEOCALL_EVENT_REPLY 事件后要根据被服务的一方（如客户）的回复情况实现不同业务逻辑处理，参考

代码如下：

```
String strMessage = null;
        switch (dwErrorCode) {
        case AnyChatDefine.BRAC_ERRORCODE_SESSION_BUSY:
            strMessage =
mContext.getString(R.string.str_returncode_bussiness);
            break;
        case AnyChatDefine.BRAC_ERRORCODE_SESSION_REFUSE:
            strMessage =
mContext.getString(R.string.str_returncode_requestrefuse);
            break;
        case AnyChatDefine.BRAC_ERRORCODE_SESSION_OFFLINE:
            strMessage = mContext.getString(R.string.str_returncode_offline);
            break;
        case AnyChatDefine.BRAC_ERRORCODE_SESSION_QUIT:
            strMessage =
mContext.getString(R.string.str_returncode_requestcancel);
            break;
        case AnyChatDefine.BRAC_ERRORCODE_SESSION_TIMEOUT:
            strMessage = mContext.getString(R.string.str_returncode_timeout);
            break;
        case AnyChatDefine.BRAC_ERRORCODE_SESSION_DISCONNECT:
            strMessage =
mContext.getString(R.string.str_returncode_disconnect);
            break;
        case AnyChatDefine.BRAC_ERRORCODE_SUCCESS:
            break;
        default:
            break;
        }
        if (strMessage != null) {
            BaseMethod.showToast(strMessage, mContext);
        }
        }
```

7.3 视频呼叫开始

被服务的一方（如客户）通过调用 VideoCallControl 接口（视频呼叫控制接口）去发送 BRAC_VIDEOCALL_EVENT_REQUEST 事件同意了请求方（如坐席）的视频呼叫请求，则系统会触发 BRAC_VIDEOCALL_EVENT_START 事件，系统会自动的分配一个房间号，双方都需要进入该房间号，程序需要响应此事件实现开始视频呼叫的业务逻辑功能。参考代码如下：

在进入房间后，Android 客户端 SDK 会发送 WM_GV_ENTERROOM 消息，程序需要捕获到此消息并进行响应。参考代码:

```
public void OnAnyChatEnterRoomMessage(int dwRoomId, int dwErrorCode) {
        //进入房间成功，打开本地音视频
        if (dwErrorCode == 0) {

            anychat.UserCameraControl(-1, 1);
            anychat.UserSpeakControl(-1, 1);
            bSelfVideoOpened = false;
        }
    }
```

7.4 视频呼叫结束

在整个视频通话服务完成后，任一方都可以通过调用 VideoCallControl 接口（视频呼叫控制接口）去发送 BRAC_VIDEOCALL_EVENT_FINISH 事件结束当前的视频呼叫。参考代码如下：

```
//发送视频呼叫回复指令，dwErrcode=0
AnyChatCoreSDK.VideoCallControl(AnyChatCoreSDK.BRAC_VIDEOCALL_EVENT_REPLY, SuserId,
0, 0, 0, "");
```

八、资源释放

(1) 离开房间

释放当前房间内的音视频资源。参考代码如下：

```
//离开指定房间，-1 表示离开当前所在房间
anychat.LeaveRoom(-1);
```

在音视频交互结束后，可调用该操作。离开当前房间之后，可再次选择进入指定房间。

(2) 退出

断开与 AnyChat 通讯服务器连接。参考代码如下：

```
//断开与服务器的连接
anychat.Logout();
```

在需要断开跟 AnyChat 服务器通讯连接的时候，可调用该操作。退出之后，可以再次调用连接、登录服务器。

(3) 释放资源

释放整个 SDK 资源。参考代码如下：

```
//释放资源
anychat.Release();
```

建议在退出整个应用程序的时候调用该操作。释放 SDK 之后，需要重新初始化 SDK 之后才能进行连接、登录、进入房间等操作。

九、附录

本附录中包括了开发流程指南中所用到的示例程序的运行截图。

9.1 HelloAnyChat 界面

AnyChat for Android SDK 包提供的 HelloAnyChat 程序（源码在“src\HelloAnyChat”目录下）最终运行效果如下图所示：

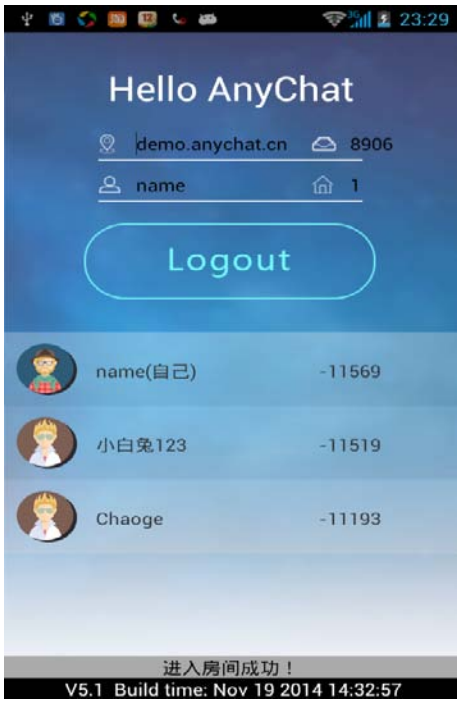


图 9-1 登录界面

图 9-2 视频通话界面

9.2 AnyChatQueue 界面

AnyChat for Android SDK 包提供的 AnyChatQueue 程序（源码在“src\AnyChatQueue”目录下）最终运行效果如下图所示：

图 9-3 登录界面

图 9-4 营业厅界面

图 9-5 队列选择界面

图 9-6 视频通话界面