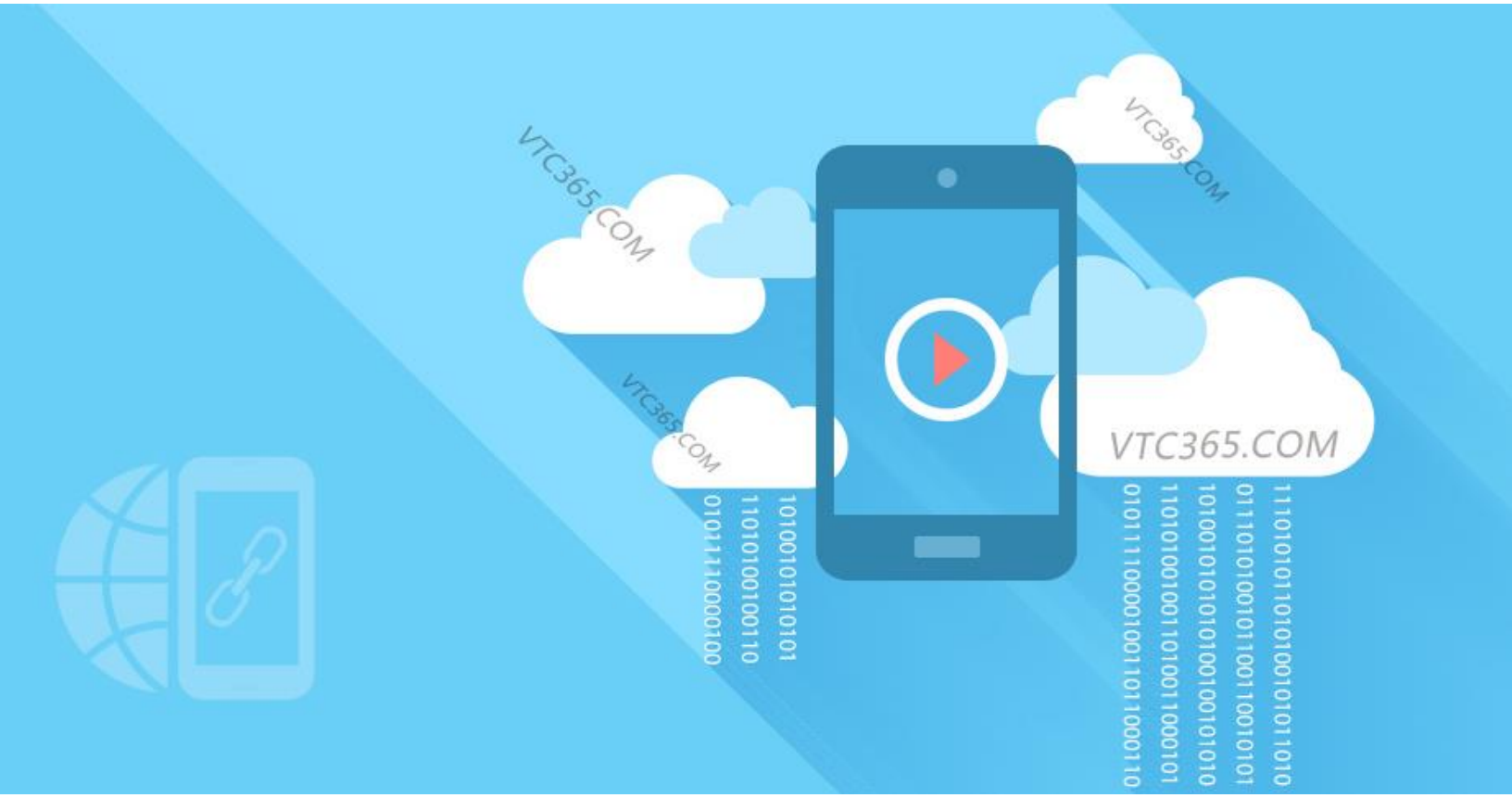



# VTC 移动可视云平台及 SDK 产品和服务

## Android 版本接口文档

发布时间：2016 年 8 月

软件版本：V2.5

南京微特喜网络科技有限公司

# VTC 移动可视云平台及 SDK 技术
# Android 版本接口文档

## 版本号：V2.5

| 文档变更记录 | | | |
|---|---|---|---|
| 版本号 | 完成日期 | 变更描述 | 作者 |
| 0.1 | 2014-12-1 | Initial version | |
| 0.2 | 2015-2-30 | Reworked version | |
| 1.0 | 2015-3-30 | First Edition version | |
| 2.0 | 2015-7-22 | Second version | |
| 2.1 | 2015-10-30 | 获取网页播放地址（3.3.2）<br>新增获取直播 URL 的方法（3.3.4） | |
| 2.2 | 2016-1-6 | 增加对接开发者已有用户系统的说明 | |
| 2.3 | 2016-3-30 | 补充对使用 vtc_chatsdk 需注册服务的说明(2.3)<br>即时通讯通知回调中增加 onGroupNotice 回调(3.2.2.4) | lh@vtc365.com |
| 2.4 | 2016-6-22 | BASE SDK 增加修改 HTTP 服务器端口的接口(3.1.1 的用户登录)<br>CHAT SDK 增加修改 XMPP 端口的接口(3.2.4.3) | lh@vtc365.com |
| 2.5 | 2016-8-2 | 即时通讯 SDK 增加重连接口(3.2.2.5)<br>直播 SDK 增加设置帧率和码率接口(3.3.2) | jjf@vtc365.com |

# 目录

# 1 引言

## 1.1 概要

VTC 移动可视云平台是一个提供 PaaS 层移动流媒体通信服务的云计算平台。平台提供 SDK 关键技术，支持视频分享、IP 电话、视频通信、视频直播等平台级移动流媒体基础功能。用户通过 SDK 技术可在其终端(APP, iPad, TV)快速实现类似微信的基础功能，并基于该基础功能开发自身产品和业务。

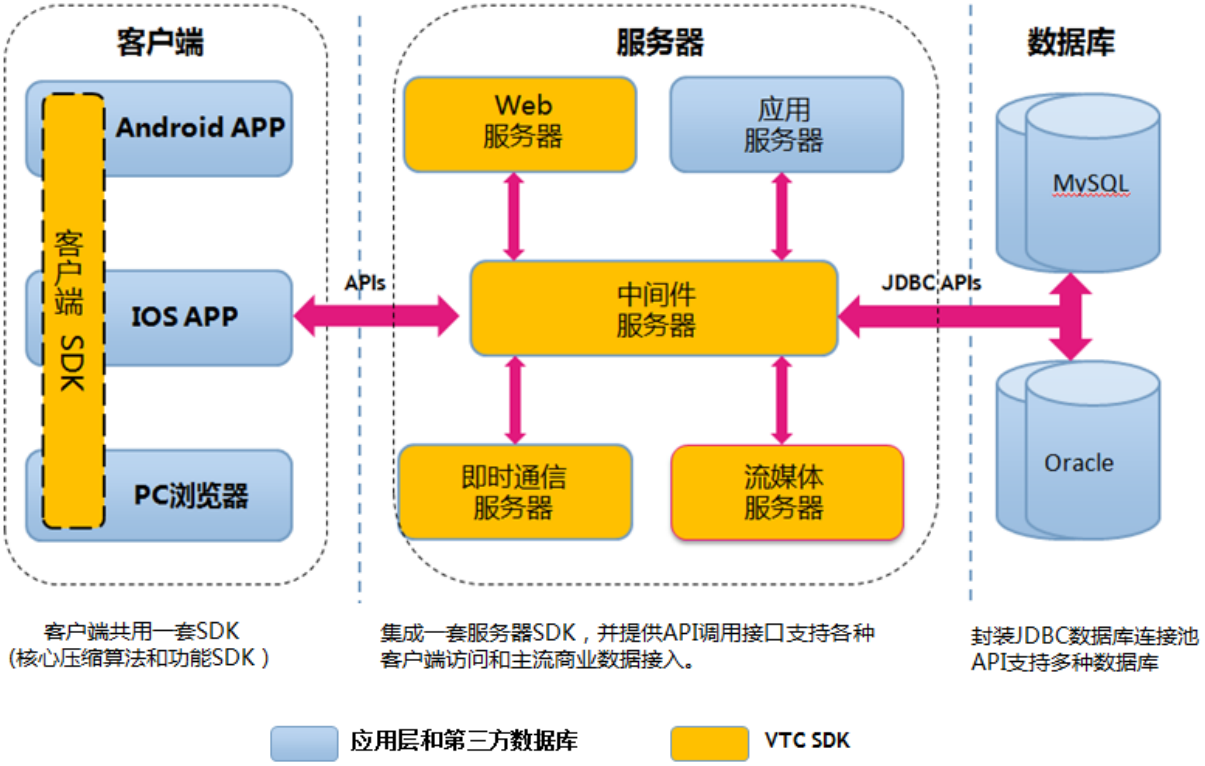
## 1.2 重要假设条件

- 用户具有基础的 Android 应用层开发能力
- 用户具备基本的 Android 代码调试、跟踪能力

# 2 运行环境

| | 硬件 | 软件 |
|---|---|---|
| 终端 | Android 手机 | OS：Android 4.0 系统以上。<br>HW：推荐 2012 年以后的双核以上手机。<br>带支持 NEON 多媒体加速指令集（伪硬件模块），"ARMv7 with NEON" or "Cortex A8/A9 "及以上 |
| | iPhone 手机 | OS: iOS-7, iOS-8<br>HW: iPhone4S, iPhone5，iPhone5S, iPhone6, iPhone6P（推荐 iPhone5s） |
| | PC 机 | OS: Windows7, WindowsXP,<br>浏览器：要求支持 IE8, Safari(on IOS), Chrome 安装 VTC 插件 |
| 服务器 | x86CPU/4 核/ 8G RAM / 720G Disk 以上配置 | Linux OS: CentOS 6.2<br>Database：MySQL, Oracle10g<br>Web Server: Apache/Tomcat7 |
| 网络 | Wi-Fi: 电信 DSL+WLAN | 流畅直播：上、下行最小带宽 〉256 kbps<br>标清直播：上、下行最小带宽〉1.2Mbps |
| | 4G: TD-LTE | |
| | 3G: CDMA-EVDO/WCDMA/ | |

## 2.1 系统架构




## 2.2 用户接入管理

通过 www.vtc365.com 注册类型为“企业用户”，获得一个 VTCXXXXX 的 UID，填写相关公司名称和营业执照加盖公章后的照片和联系人资料，注册提交后由人工审核。

## 2.3 SDK 模块名称

SDK 包含 4 个模块包：

vtc_basesdk.jar　　提供用户注册，登录和文件上传下载

vtc_chatsdk.jar　　提供即时通讯功能，依赖 vtc_basesdk.jar

vtc_livesdk.jar　　提供视频录制，直播和播放，依赖 vtc_basesdk.jar

vtc_talksdk.jar　　提供音视频通信，依赖 vtc_basesdk.jar，如果需要使用此通讯 sdk，需要在 AndroidManifest.xml 的配置文件中注册一个 Service：

```
<service
android:name="com.vtc365.api.XMPPservice"
```

```
android:enabled="true">
</service>
<receiverandroid:name="com.vtc365.api.VtcReceiver">
<intent-filter>
<actionandroid:name="android.net.conn.CONNECTIVITY_CHANGE"/>
</intent-filter>
</receiver>
```

# 3 SDK 接口规范

SDK 接口分为客户端接口和服务器接口。客户端接口分别为 APP 上层应用之间的接口（外部接口）和 APP 与服务器间接口（内部接口），目前支持 Android，IOS,和 Web JS 接口，本文档为 Android SDK 接口。服务器接口为与客户端和与客户业务服务器之间的接口，主要体现为 HTTP 接口，将在单独的服务器文档中描述。

Android SDK 包含一系列的 jar 包文件和*.so 运行库文件。使用者需要先导入响应的 jar 和 so 文件到应用工程的 libs 目录。

## 3.1 BASE SDK

### *3.1.1 BASE SDK API 接口对象*

1）创建对象

应用调用 BASE SDK 完成注册，登录等功能的接口对象是 com.vtc365.api.BaseApi。应用首先要创建一个全局的 BaseApi 对象，绝大多数 BASE SDK 调用都包含在该对象中：

```
BaseApi baseApi = BaseApi.getInstance(context, appid);
```

其中 context 参数一般可以使用 Application 的 Context.

参数 appid 是用户应用 ID，如果需要接入公有云，用户需要申请应用 ID 和认证码。如果是私有云，则无需此参数。

2）注册用户

创建 BaseApi 对象后，应用可以调用 **register** 函数来创建用户。

```
publicvoid register(String userId,String password, String HOST_NAME,
   String appid, String apptoken, RegisterCallback cb, int tag);
```

其中：

userId 是用户在公有云中的用户 ID，该 ID 在同一个 appid 不能重复。

password 是用户在 VTC 公有云上的密码。

HOST_NAME 是 VTC 公有云的地址（www.vtc365.com）。

Appid 和 apptoken 是用户从 VTC 申请的应用 ID 和认证码。

cb 是用于接收注册结果的回调函数。

tag 是用户自定义的值，可以传递到回调函数中去。

【如何对接开发者已有的用户系统？】

如果开发者已经有自己的用户系统，可以使用 register 函数为用户在 VTC 公有云上创建映射的账号。注册使用的 userId 可以用该用户在开发者用户系统中的唯一标识符，password 推荐使用开发者用户系统中密码的 HASH 值，这样既不会泄漏用户密码，又不用保存一个额外的公有云的密码。

3）用户登录

应用需要调用 login 函数来登陆系统：

```
publicvoidlogin(String user, String password, String host);
```

user ->用户 id 或者昵称

password ->用户密码

host -> VTC 服务器域名或者 IP，例如 www.vtc365.com

若需要修改 HTTP 服务器的端口，则需调用以下 setServerPort 函数：

```
public void setServerPort(String serverPort);
```

serverPort -> HTTP 服务器端口

## *3.1.2 BASE SDK 回调接口*

SDK 通过回调来给应用返回系统通知或者操作结果。应用需要实现这些接口，并传递给 SDK。

### 3.1.2.1 登录状态

```
publicinterface LoginCallback {
        publicvoid onStatus(int result);
}
```

调用者实现改接口并用 baseApi.setLoginCallback()注册。登录服务器过程中会收到 onStatus()调用，result 参数表示结果：

| “result”值 | 意义 |
|---|---|
| *LOGIN_HTTP_FAILED* | 登录失败(网络错等) |
| *LOGIN_HTTP_INVALID* | 用户名密码错 |
| *LOGIN_HTTP_NOT_FOUND* | 该用户不存在，应用可以使用 register()方法发起注册。对于存量用户，建议使用公有云服务器 API 一次性创建。对于新用户，可以在注册自有服务器时同时注册公有云用户。 |

### 3.1.2.2 注册结果

```
publicinterfaceRegisterCallback {
        publicvoid onResult(int status, String arg, int tag);
}
```

该接口用来返回注册结果,status 有如下定义：

```
publicfinalstaticintREGISTER_OK = 0;
publicfinalstaticintREGISTER_FAILED = 1;
```

#### 3.1.2.3 异步操作结果

有些操作是长耗时的，会放到独立线程执行，此接口用于获取结果。

```
publicinterface OperateCallback {
	publicvoid onSuccess();
	publicvoid onFailure(String reason);
}
```

#### 3.1.2.4 需要上传文件的异步调用接口

```
publicinterface OperateFileCallback {
	publicvoid onSuccess(Map<String, String> params);
	publicvoid onFailure(String reason);
	publicvoid onProgress(int param1, int param2);
}
```

#### 3.1.2.5 获取用户信息回调接口

```
publicinterface GetUserInfoCallback {
	publicvoid onSuccess(Map<String, String> info);

	publicvoid onFailure(String error);
}
```

用来返回用户信息，在 onSuccess 函数回调中，info 包含用户的信息，可以用“nickname”作为关键字获取昵称，用“headIcon”作为关键字获取头像。

#### 3.1.2.6 查找用户回调接口

```
publicinterface SearchUsersCallback {
	publicvoid onSuccess(List<UserItem> result, int pages);
	publicvoid onFailure();
}
```

此接口用来返回查找用户的结果。

### *3.1.3 BASE SDK APIs*

#### 3.1.3.1 获取登录用户信息

每个用户在 VTC 公有云系统中均会分配一个数字编号，即 VTC ID。在即时通讯，视频直播，音视频通信中均需要使用该 ID，BASE SDK 提供 API 来获取：

获取 ID：

```
public String getMyId();
```

获取头像地址：

```
public String getMyPortrait();
```

获取登录用户昵称：

```
public String getMyNick();
```

获取音视频所需要的STUN/TURN服务器：

```
public String getStunServer();
public String getTurnServer();
```

头像地址是服务器上的相对地址，下载时需要以此作为前缀：

```
public String getBaseURL();
```

#### 3.1.3.2 好友操作

加好友关注，如果同时使用即时通讯SDK，对方会收到通知：

```
publicvoid requestAddFriend(String user, OperateCallback cb);
```

取消好友关注：

```
publicvoid requestRemoveFriend(String id, OperateCallback cb);
```

同意加好友，双方会出现在即时通讯的通讯录中：

```
publicvoid agreeAddFriend(String user, OperateCallback cb);
```

拒绝加好友：

```
publicvoid refuseAddFriend(String user, OperateCallback cb);
```

#### 3.1.3.3 获取用户信息

```
publicvoid getUserDetails(String id, final GetUserInfoCallback
cb);
```

#### 3.1.3.4 修改登录用户信息

修改昵称：

```
publicvoidchangeMyNick(final String nick, final OperateCallback cb);
```

上传新头像：

```
publicvoid changeMyPortrait(String path, OperateCallback cb);
```

#### 3.1.3.5 下载上传

下载文件：

```
publicvoid downloadFile(String url, boolean noBaseUrl, String
pathToSave, BaseApi.OperateFileCallback callback);
```

其中noBaseUrl为true，会自动加上服务器baseURL

上传文件：

```
publicvoid uploadFile(int type, String path, OperateFileCallback cb);
```

type表明上传分享的类型，有如下定义：

BaseApi.UPLOAD_IMAGE 图像文件
BaseApi.UPLOAD_VOICE 音频文件
BaseApi.UPLOAD_VIDEO 视频文件

#### 3.1.3.6 登出

```
publicvoid logout();
```

#### 3.1.3.7 查找用户

```
publicvoid searchUsers(String keyword, int page, SearchUsersCallback cb);
```

此API可以根据关键字查找用户。

参数中keyword是查找关键字。

查找结果每8个分为一页，page参数控制返回结果的起始位置，比如说1就表明需要查找前8个，是2就表示查找9-16个用户，以此类推。

查找是异步进行，结果在回调接口cb中返回。

## 3.2 即时通讯 SDK

### *3.2.1 即时通讯 SDK 接口对象*

即时通讯 SDK 接口对象是 com.vtc365.api.ChatApi。要调用即时通讯 SDK 需要先获取此对象：

ChatApi chatApi = ChatApi.getInstance(context, baseApi);

context 是应用的 Context

baseApi 是 BaseSDK 接口对象

### *3.2.2 即时通讯回调接口*

#### 3.2.2.1 即时通讯登录回调

此接口用于获取登录状态通知，用户实现该接口并用 chatApi.setLoginCallback()来注册到 SDK 中。

```
publicinterfaceLoginCallback {
    publicvoid onStatus(int result); /* 登录状态 */
}
```

onStatus()会汇报 XMPP 登录状态，其中 status 参数定义如下：

```
publicfinalstaticint LOGIN_XMPP_OK = 0; /* 登录成功 */
publicfinalstaticint LOGIN_XMPP_CONFLICT = 1; /* 登录冲突被踢出服务器(同一帐号多次登录) */
```

```
publicfinalstaticintLOGIN_XMPP_CLOSED = 2; /* 登录失败(网络错或密码错) */
```

#### 3.2.2.2 即时通讯通讯录回调

此接口用于获取通讯录通知，用户实现该接口并用 chatApi.setRosterCallback()来注册到 SDK 中。

```
publicinterfaceRosterCallback {
    publicvoid onRosterChanged(boolean entries, boolean presence); /* 通讯录变化 */
}
```

#### 3.2.2.3 即时通讯新消息回调

此接口用于获取原始消息通知，对于需要实现自己消息扩展的用户推荐实现此接口并用 chatApi.setRawMessageCallback()注册到 SDK 中。

```
publicinterface RawMessageCallback {
    publicvoid onMessage(ChatRawMessage incoming);
    publicvoid onControlMessage(ChatRawMessage incoming);
}
```

此接口用于获取VTC的扩展消息，支持文本图像语音和视频，用户实现该接口并用 chatApi.setMessageCallback()注册到SDK中来接收：

```
publicinterface MessageCallback {
    publicvoid onMessage(ChatMessage incoming); /* 新消息(同时会保存数据库) */
}
```

#### 3.2.2.4 即时通讯通知回调

应用可以用此回调接口来获得系统通知消息：

```
publicinterfaceNoticeCallback {
    publicvoid onGroupRequest(String from, String group, int code,long timestamp);/* 群请求 */
    publicvoid onSystemMessage(String from, String text, int msgflag); /* 系统通知 */
    publicvoid onFriendRequest(String from, int code, int arg); /* 好友通知 */
    publicvoid onLiveVideo(String from, LiveSession session,
                boolean offline, Calendar delay); /* 直播通知 */
    publicvoid onLiveAck(String from, String resp); /* 直播回应 */
    publicvoid onLiveVideoStop(String from); /* 直播停止通知 */

    public void onGroupNotice(String originalJsonString, String groupName, String userName, String operator, String action);/* 群成员变
```

```
动 */

    }
```

其中 onGroupRequest() 会汇报群请求，code 的定义如下：

```
publicfinalstaticintJOIN_GROUP_REQUEST = 0; /* 加群请求 */
publicfinalstaticintJOIN_GROUP_AGREED = 1; /* 同意 */
publicfinalstaticintJOIN_GROUP_REFUSED = 2; /* 群主同意 */
publicfinalstaticintGROUP_MEMBER_REMOVED = 3; /* 群主拒绝 */
```

onFriendRequest() 汇报好友请求,code 定义如下：

```
publicfinalstaticintADD_FRIEND_REQUEST = 4;
publicfinalstaticintREMOVE_FRIEND_REQUEST = 5;
publicfinalstaticintADD_FRIEND_REFUSED = 6;
publicfinalstaticintADD_FRIEND_AGREED = 7;
```

onGroupNotice()群成员变动，当群中成员发生变化时，在该回调用可以做刷新成员列表、群列表等操作，避免群成员在加入/退出时成员列表或群列表不更新的现象，回调中各参数含义如下：

| 名字 | 用处 |
|---|---|
| originalJsonString | 来自通讯服务器的原始信息的 json 字符串 |
| groupName | 成员加入群的 group_id |
| userName | 与群关系发生变动的成员的 user_id |
| operator | 增删群成员的管理员的 user_id |
| action | 成员变动的情况，值有 5 种：MemberBatchAdd(管理员批量增加成员)、MemberDelete(管理员删除成员)、MemberApprovedJoin(管理员同意成员加入)、MemberQuit(成员主动退出群)和 GroupDelete(群被删除) |

#### 3.2.2.5 即时通讯重连回调

在即时通讯 SDK 和服务器的连接中断需要重新连接时，使用此接口通知。当用户实现此接口并注册到 SDK 中后，可以控制重连的行为：

```
public interface ReConnectCallback {
      public boolean isAllowed();
}
```

当 `isAllowed()`返回 `true` 时，允许重连，否则 SDK 将中断重连。用户可以重新登出再登录来恢复。此接口可以用来实现断线登录前，先判断用户是否允许在此设备登录，如是否已经被其它设备踢出登录。

### *3.2.3 即时通讯 SDK 数据对象*

SDK 用不同对象来传递信息，比如聊天消息，好友消息等。

#### 3.2.3.1 好友记录 ChatFriend

com.vtc365.api.ChatFriend 包含如下 getters/setters：

| 名字 | 用处 |
|---|---|
| getFriendId()/setFriendId() | 好友的 ID |
| getNickName()/setNickName() | 昵称 |
| getHeadIcon()/setHeadIcon() | 头像 URL |

#### 3.2.3.2 群记录

com.vtc365.api.ChatGroup 包含如下 getters/setters：

| 名字 | 用处 |
|---|---|
| getGroupId()/setGroupId() | 群的 ID(系统唯一的) |
| getGroupName()/setGroupName() | 群自定义名字 |
| getHeadIcon()/setHeadIcon() | 群头像 URL，同 ChatFriend |
| getGroupOwner()/setGroupOwner() | 群主 ID |
| getMemberCount()/setMemberCount() | 群成员数目 |

#### 3.2.3.3 群成员记录

com.vtc365.api.ChatGroupMember 包含：

| 名字 | 用处 |
|---|---|
| getGroupId()/setGroupId() | 群 ID |
| getMemberId()/setMemberId() | 成员用户 ID |
| getNickName()/setNickName() | 昵称 |
| getHeadIcon()/setHeadIcon() | 头像 |
| getRole()/setRole() | 角色，如”admin”,”member” |

#### 3.2.3.4 消息记录

com.vtc365.api.ChatMessage 包含：

| 名字 | 用处 |
|---|---|
| getType()/setType() | 消息类型：<br>ChatApi.MESSAGE_TYPE_TEXT<br>ChatApi.MESSAGE_TYPE_IMAGE<br>ChatApi.MESSAGE_TYPE_VOICE<br>ChatApi.MESSAGE_TYPE_VIDEO<br>ChatApi.MESSAGE_TYPE_LOCATION |
| getIsGroup()/setIsGroup() | 是否群消息 |
| getFromId()/setFromId() | 发送方 ID,如果是收到的群消息则为群 ID |
| getToId()/setToId() | 接收方 ID,如果是发送的群消息则为群 ID |
| getTimestamp()/setTimestamp() | 时间 |
| getBody()/setBody() | 消息体,文本字符串 |
| getNick()/setNick() | 发送方的昵称 |
| getHeadIcon()/setHeadIcon | 发送方的头像 |
| getURL()/getPreviewURL()/getLocalPath() | 用于从消息体解析出图像，声音和视频的 URL,预览图 URL,和发送方的本地文件路径 |
| getLng()/setLng() | 发送消息时客户的经度 |
| getLat()/setLat() | 发送消息时的纬度 |
| getLocationLng() | 从消息体解析出 ApiClient.MESSAGE_TYPE_LOCATION_类型 |

| | |
|---|---|
| | 消息携带位置的经度 |
| getLocationLat() | 从消息体解析出 ApiClient.MESSAGE_TYPE_LOCATION 类型消息的纬度 |

此消息记录是 VTC 实现的消息扩展，支持音频视频和图像消息，如果应用自己实现，可以使用 ChatRawMessage 消息记录。

com.vtc365.api.ChatRawMessage 包含：

| 名字 | 用处 |
|---|---|
| getStatus() | 消息状态, 0 是发送消息, 3 是接收消息 |
| getIsGroup() | 是否群消息 |
| getFromId() | 发送者 |
| getToId() | 接受者 |
| getTimestamp() | 消息时间戳 |
| getDelay() | 离线消息时间 |
| getBody() | 消息体 |
| getProperties() | 消息附带参数 |

## *3.2.4 即时通讯 SDK APIs*

### 3.2.4.1 设置回调

```
publicvoidsetCallback(ChatApiCallback cb);
publicvoid setVtcCallback(VtcChatApiCallback cb);
publicvoid setReconnectCallback(ReConnectCallback cb);
```

### 3.2.4.2 服务器连接状态

```
publicbooleanisXmppAvailable();
```

### 3.2.4.3 登录登出

```
publicvoid login(String user, String password, String host);
publicvoid logout();
```

若需要修改XMPP端口，则需调用setXmppPort函数修改：

```
public void setXmppPort(int xmpp_port);
```

### 3.2.4.4 获取好友列表

```
public List<ChatFriend> getFriendsFromRoster();
```

### 3.2.4.5 从网路请求刷新好友

```
publicvoid refreshFriends();
```

调用者注册的RosterCallback()会被调用。

#### 3.2.4.6 建群

```
voidcreateGroup(String name, String type, String headIcon, String nick, final
OperateCallback cb);
```

name 是群ID需要唯一否则会创建失败
headIcon 是群的头像URL
nick 是群的名字

当前登录用户会成为管理员

#### 3.2.4.7 群管理员加用户

```
void createGroupMembers(String group, List<String> ids, final OperateCallback
cb);
```
ids里面包含需要批量加入群的用户id

#### 3.2.4.8 申请加入群

```
void joinGroup(String group, final OperateCallback cb);
```
管理员会在NoticeCallback.onGroupNotice()收到通知。

#### 3.2.4.9 群管理员同意加入群

```
void agreeJoinGroup(String group, String user, final OperateCallback cb);
```

#### 3.2.4.10 群管理员拒绝加入群

```
void refuseJoinGroup(String group, String user, final OperateCallback cb);
```

#### 3.2.4.11 删除群

```
void deleteGroup(String group, final OperateCallback cb);
```

#### 3.2.4.12 群成员退出群

```
voidquitGroup(String group, final OperateCallback cb);
```

#### 3.2.4.13 发送单聊消息

```
publicboolean sendMessage(String to, String body,
                Map<String, String> properties);
```

```
publicbooleansendMessage(int type, String to, String body, String nick, int
duration, String headIcon, OperateFileCallback client, double lng, double
lat);
```

| 参数 | 用处 |
|---|---|
| properties | 自定义扩展 |
| Type | ApiClient.MESSAGE_TYPE_TEXT<br>ApiClient.MESSAGE_TYPE_IMAGE<br>ApiClient.MESSAGE_TYPE_VOICE<br>ApiClient.MESSAGE_TYPE_VIDEO |

| | |
|---|---|
| To | 对方的ID |
| Body | 消息体字符串，对于图片视频和语音，是本地文件路径。 |
| Nick | 发送人昵称 |
| Duration | 语音长度（秒） |
| headIcon | 发送人头像 |
| Client | 对于发送ApiClient.MESSAGE_TYPE_TEXT忽略，对于其它消息用来返回发送进度和结果。 |
| Lng/lat | 经纬度 |

#### 3.2.4.14 发送群聊

```
publicboolean sendGroupMessage(String to, String body,
                Map<String, String> properties);
publicbooleansendGroupMessage(int type, String to, String body, String nick,
int duration, String headIcon, OperateFileCallback client, double lng, double
lat);
```

| | |
|---|---|
| Nick | 群昵称 |
| headIcon | 群头像 |

#### 3.2.4.15 发送位置

```
publicboolean sendLocation(String to, boolean isGroup, String path, String
nick, String headIcon, double lng, double lat,
                OperateFileCallback client, double my_lng, double my_lat);
```

| | |
|---|---|
| Path | 位置在地图上的截图 |
| Lng | 位置对应的经度 |
| Lat | 位置对应的纬度 |
| My_lng/my_lat | 发送消息时的经纬度 |

#### 3.2.4.16 转发消息

```
publicboolean forwardMessage(ChatMessage msg, String to, boolean isGroup,
String nick, String headIcon, double lng, double lat);
```

## 3.3 音视频分享和直播 SDK

### *3.3.1 音视频直播分享 SDK 初始化*

LiveApi live = LiveApi.getInstance(context, BaseApi baseApi);

### 3.3.2 调用VTC 高效音视频录制或直播

在录制或者直播的 Activity 里面，应用需要创建 SDK 中的 com.vtc365.LiveBroadcast 对象，录制和直播的 API 封装在该对象中：

```
public LiveBroadcast(Context ctx, ApiClient api, Handler handler, int mode, Vector<String> targets);
```

其中 api 是前面创建的 ApiClient 对象

其中handler用来接收直播处理成功或失败的消息,msg.what定义为:

LiveBroadcast.START_SUCCESS

LiveBroadcast.START_FAILED

参数 mode 是决定本次是网络直播还是本地录制视频:

LiveBroadcast.LIVING

LiveBroadcast.LOCAL

参数 targets 是直播目标的ID数组，直播通知将发给这些目标用户。如果不需要通知，可以传入一个空的Vector<String>对象(不是null)。

创建 LiveBoradcast 对象后，应用要把录制和直播窗口放到自己的界面中:

```
publicvoid attach(ViewManager wm);
```

用户可以调用如下函数来决定直播是否本地留记录（在SD卡/vtcVideo目录），缺省是保留:

```
publicvoid setKeepLocalCopy(boolean keepLocalCopy);
```

用户可以通过如下函数获得摄像头图像大小（返回如果是空，则还没有开始预览，调用者需要延时然后再次调用）:

```
publicSize getVideoSize();
```

然后调用如下函数开始录制或直播(如果是直播，接收方会收到通知):

```
publicvoid start();
```

直播SDK设置了缺省的视频的参数，如果用户有修改缺省参数的需求，可以通过以下API实现:

public void setFps(int fps)；用户可以设置采集图像的帧率，建议不要设置的太大或者太小，可以设置在10-30之间。值越大则画面越连贯，但是会增大数据量和CPU使用率。

public void setVideoBps(int bps);用户可以设置视频码率(单位Kbps)，码率越大，则视频越清晰，但是数据量会越大。可以在500到1200之间设置一个适当的值。

直播或录制中的暂停和恢复:

```
public void pause();
public void resume();
```

直播或录制停止:

```
public void stop();
```

从界面中移除:

```
public void detach();
```

直播或录制中开启闪光灯：

```
publicvoid swithFlash(boolean on);
```

直播或录制中切换摄像头：

```
publicboolean switchCamera();
```

录制结束后，可以用如下API获得录制出的mp4文件路径：

```
public String getMp4();
```

获取用于播放的RTSP地址：

```
public String getRtsp();
```

获取网页播放地址：

```
public String getUrl();
```

获取直播结束后的历史视频地址：

```
public String getOfflineUrl();
```

在收到 LiveBroadcast.START_SUCCESS 消息后,可以通过如下 API 获得该视频的观看地址，应用可以自己传递给接收者：

```
public String getRtsp();  // 获得RTSP观看地址，可以用3.5.3描述的播放器或者标准RTSP播放器观看

public String getUrl(); // 获得浏览器观看地址
```

### *3.3.3 调用系统默认音视频录制接口*

请参考Android开发文档，自行实现。
[注]：调用系统默认音视频录制接口生成的音视频文件size较大，在聊天session中分享和上传耗时和占用带宽较大，但清晰度稍好。

### *3.3.4 观看直播或观看历史视频*

收到直播通知或者分享的 mp4 文件 URL 后，可以用 SDK 来播放。在播放的 Activity 里面首先要创建 com.vtc365.VtcPlayer 对象，播放功能集成在该对象中：

```
publicVtcPlayer(Context ctx, String author, String userId, String date,
            String videoValue, int id);
public VtcPlayer(Context ctx, String url, int id);
```

第一种用于观看直播，参数可以从直播通知里面LiveSession中获取：
session.getUserId(), session.getDate(), session.getVideoValue().
同时直播通知LiveSession还可以取得直播停止后的历史视频播放地址：
session.getOfflineUrl()

第二种用于观看历史视频mp4， url 是mp4文件路径或者http URL。
Id参数用于在播放过程中出现需要缓冲的情况下显示的对话框的resource id。

创建成功后，应用需要把播放窗口放到界面上：

```
publicvoid attach(ViewManager wm);
```

然后通知窗口的大小，播放器会自动调整内容以适应窗口：

```
publicvoid setViewSize(int width, int height);
```

停止观看，在结束 Activity 之前，应用调用一下 API 来停止播放：

```
publicvoid detach();
```

对于mp4分享视频，可以通过下面的API获得视频长度和当前播放位置(秒)：

```
publicdouble duartion();
publicdouble position();
```

## 3.4 音视频通信 SDK

音视频通信 API 集成在 com.vtc365.api.SipTalk 或者 com.vtc365.api.XmppTalk 对象内，实现支持基于 SIP 或者 XMPP 的音视频通信。该对象通过 SipTalk.getInstance()或者 XmppTalk.getInstance()来获得。

对于使用 SIP 用户首先需要登录到服务器：

```
publicvoid login(String domain, String user, String pwd);
```

其中的domain是www.vtc365.com，而user是用户ID,pwd是用户密码。

对于使用XMPP的用户则需要实现XmppTalk.ISendXmpp接口，在这个接口内实现XMPP消息发送（通过即时通讯SDK），并用XmppTalk.getInstance().setConnection(sendXmpp)注册。并且对于即时通讯SDK收到的控制消息，需要调用XmppTalk.handleControlMessage()来处理。详细参见Demo代码的HuiDongApplication.java。

### *3.4.1 呼入通知*

```
publicinterfaceSignalingCallback {
    publicvoid onIncomingOffer(String user, String sdp, TalkMode
mode,
                    Calendar time);
}
```

用户应用实现 SipTalk 或 XmppTalk 的 SignalingCallback 来接收呼入通知。应用通过调用 SipTalk.getInstance().register(callback)或者 XmppTalk.getInstance().register(callback)来注册该调用接口。

### 3.4.2 视频 View

如果请求视频电话，应用需要创建 View 来呈现视频。SDK 提供 com.vtc365.view.VideoStreamsView 对象。应用在启动视频通话前需要创建改对象并放到界面适当的地方：

```
VideoStreamsView view = new VideoStreamsView(this, new Point(
this.getWindowManager().getDefaultDisplay().getWidth(),
this.getWindowManager().getDefaultDisplay().getWidth());

layout.addView(videoStgreamsView);
```

### 3.4.3 被叫方

应用会在收到 IP 电话和视频电话请求时收到回调请求。此时应用可以在界面上显示电话通知，当用户点击接听按钮时，应用调用 API 来建立呼叫：

```
publicbooleanstartTerm(Context context, Handler ui,
        VideoStreamsView view, int vkbps,
        String sdpOffer);
```

其中 view 为视频电话呈现窗口，IP 电话时可以为 null。
参数 vkbps 控制视频占用的目标带宽(Kbps)，当设为 0 时将由系统自动选择。(目标带宽为理想值，实际运行中，系统会根据 CPU 和网络情况自动调整)
参数 sdpOffer 为新电话通知里面传入的参数。

Handler 用来接收 IP 和视频电话的通知，目前有如下通知：

| 通知 (msg.what) | 意义 |
|---|---|
| SipTalk.TALK_STARTED | 呼叫建立 |
| SipTalk.TALK_FAILED | 呼叫处理出错 |
| SipTalk.TALK_ANSWERED | 对方接听(主叫方会收到) |
| SipTalk.TALK_HANGUP | 对方挂断 |
| SipTalk.TALK_PREPARED | 呼叫发出，等待处理 |
| SipTalk.TALK_REFUSED | 对方拒绝(主叫方会收到) |
| SipTalk.TALK_BUSY | 对方忙(主叫方会收到) |
| SipTalk.TALK_VIDEO_OFF | 对方关闭了视频，收到本消息后应用可以用 talkVideoOff()来关闭本端的视频(限于 XMPP 作为通话信令) |
| SipTalk.TALK_BANDWIDTH_WARN | 视频通信中提示网络带宽是否能满足要求。此消息的 arg1 参数为 0 表示带宽正常，为 1 表示带宽不够，建议关闭视频。此通知会周期性发送(视频关闭前)。 |

对于在用户选择接听之前（调用 answerTalk 之前）,应用可以先用如下方法注册一个 Handler 用来处理可能的对方提前挂断的通知：

```
publicvoidsetHandler(Handler ui);
```

如果被叫方用户拒绝通话，则可以调用如下函数来结束：

```
publicvoidrefuse();
```

### 3.4.4 主叫方

主动发起 IP 或视频电话使用 API 函数：

```
publicboolean startOrig(Context context, Handler ui,
                VideoStreamsView view, int vkbps, TalkMode mode,
String userId);
```

其中 userId 是对方的 ID，mode 是通话类型，有如下选择：

TalkMode.MODE_VOICE (语音 IP 电话)和 TalkMode.MODE_VIDEO(视频电话)

其它参数同 3.4.3.

### 3.4.5 通话中的控制

在通话过程中 Activity 被切换后台，应用需要调用如下 API 来暂停和恢复视频流：

```
publicvoid pause();
publicvoid resume();
```

麦克风静音：

```
publicvoidmuteMic(boolean mute);
```

打开扬声器：

```
publicvoidsetSpeakerOn(boolean on);
```

判断能否切换摄像头：

```
publicbooleancanSwitchCamera();
```

视频通话当前摄像头(返回 front 和 back)：

```
publicStringgetCurrentCamera();
```

切换视频通话摄像头：

```
publicvoidswitchCamera();
```

结束通话需要调用下函数并调用 endCall()：

```
publicvoidsendBye(id, SipTalk.TALK_HANGUP);
```

# 4 数据库对接

详见服务器 SDK 接口文档

# 5 版本升级

## 5.1 兼容性

Android 系统不同厂家定制的碎片化，本 SDK 目前已支持华为、小米、三星、HTC、联想等主流机型。

当客户端操作系统版本升级后，客户端 SDK 有时可能需要针对少数 API 的升级优化。公有云独立节点运营客户和私有云用户将优先保证及时升级，公有云用户 SDK 将在统一时间通知发布更新版本。

当服务器版本经过 2 个大版本(1.x.y -> 2.x.y -> 3.x.y)升级后，客户端 SDK 需要同步升级以匹配服务器功能。对相差两个大版本的客户端 SDK 本系统不再保证 100%支持。

## 5.2 互联互通

Android 与 IOS SDK 之间跨平台通信已通过服务器标准 XMPP 和 SIP 信令协议支持。

# 6 Glossary 词汇表

## 6.1 ACRONYMS 首字母缩写

| Acronym | Expansion |
|---|---|
| VTC | Video Technologies Communications |
| SDK | Software Development Kit |
| API | Application Interface |
| XMPP | The Extensible Messaging and Presence Protocol |
| SIP | Session Initiation Protocol |
| HLS | HTTP Live Streaming protocol |
| RTP | Real Time Protocol |
| RTSP | Real Time Streaming Protocol |
| SVN | Subversion management tool |