

移动直播 解决方案 产品文档



腾讯云

【版权声明】

©2013-2018 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

解决方案

- 单场次直播 (PGC)
- 自由开播 (UGC+OGC)
- 弹幕聊天室 (IM)
- 录制回看 (VOD)
- 直播连麦 (LiveRoom)
- 主播 PK (LiveRoom)
- 视频通话 (RTCRoom)
- 直播答题 (冲顶模式)

解决方案

单场次直播 (PGC)

最近更新时间：2018-07-11 12:08:12

单场次直播是指同时只有一条（或几条）直播流进行直播，直播内容都是官方 PGC 内容，常被客户用于活动直播、领导讲话等直播场景，2017年底非常红火的冲顶在线答题也是单场次直播的典型应用场景。

单场次直播的接入非常简单，由于大都是PGC资源，所以主播（推流）端都有比较成熟的解决方案，您的研发资源主要是实现直播流在各个终端的音视频播放：

推流和播放URL

单场次直播，同时直播的流不会太多，所以使用 [直播管理控制台](#)>>[接入管理](#)>>[直播码接入](#)>>[地址生成器](#) 手动生成推流和播放 URL 即可，这一部分的详细介绍可以参考文档 [DOC](#)。

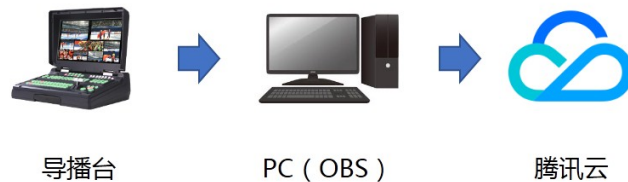
需要注意的是两点，这是以往刚对接腾讯云的客户端反复问及的问题：[为什么推不上去？](#)

- 过期时间不要太短，超过过期时间的 URL 即视为无效。

- 一个推流 URL 不可以有两个人同时推流，否则会相互影响。

如何推流

case1: 导播台推流



适用于比较正式的直播场景，采用专业导播设备对接 PC 台式机，然后通过 PC 上的 [Obs Studio](#) 进行推流。

- **方案优势：**由于导播台的存在，直播过程中切换广告或者插播其它视频都会比较容易，2017年底非常红火的冲顶在线答题均是采用这种解决方案。
- **参考文档：**Obs 推流说明可以参考 [DOC](#)。

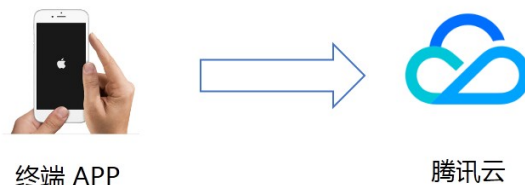
case2: 摄像机推流



适用于 **活动直播**、**现场直播** 等无法在演播室实现的直播场景，摄像机可以通过 HDMI 线直连笔记本电脑，然后用 [Obs Studio](#) 进行推流，也可以直连视频编码盒，通过视频编码盒进行推流。

- **方案优势：**不受直播地点的限制。
- **参考文档：**Obs 推流说明可以参考 [DOC](#)。
- **特别注意：**直播对网络质量要求是比较高的，但活动直播或是现场直播，现场的网络情况可能并不是非常确定，请务必提前做好网络质量测试和演练，并在直播现场确保 WiFi 和 4G 双线路并存，有网线是最好的。

case3: 移动端推流



由于近些年手机端性能的突飞猛进，目前手机直播在实际画面效果上，一点也不逊于前两种解决方案。您可以找一台性能较好的手机，安装[视频云工具包](#)，并使用其中的 **RTMP 推流** 即可直接开始推流。

- **方案优势**：简单易用、快速上手、启动成本低。
- **参考文档**：如果您想要在直播流中打入自己的水印，可以在 SDK 下载开发包，自行替换 Demo 里的水印，推流 SDK 的开发文档请参考（[iOS](#) | [Android](#)）。
- **特别注意**：腾讯云视频工具包在推流到腾讯云时采用的是抗丢包能力和带宽拼抢能力更强的 UDP 协议，在推其他云商时采用的是常规 RTMP 协议，所以推流到腾讯云，视频的流畅度会更好。

如何播放

iOS 端播放器

- **step1**: 下载腾讯云 [SDK](#) 开发工具包，如果不需要推流，可以下载独立播放器版本。
- **step2**: 参考文档 [TXLivePlayer](#) 将 SDK 集成到您的 SDK 中。

Android 端播放器

- **step1**: 下载腾讯云 [SDK](#) 开发工具包，如果不需要推流，可以下载独立播放器版本。
- **step2**: 参考文档 [TXLivePlayer](#) 将 SDK 集成到您的 SDK 中。

Web 端播放器

- **对接指引**：由于 Web 页面可以直接引用 javascript 组件，所以您只需要参考文档 [TCPlayer](#) 对接 Web 播放器即可。
- **延时很高**：Web 播放器播放 HLS（m3u8）协议的 URL 在各种终端上的兼容性比较好，但是相比于 FLV 协议（延迟在 2s-5s 左右）延迟要大很多（HLS 协议的延迟一般都在 20s 以上）。

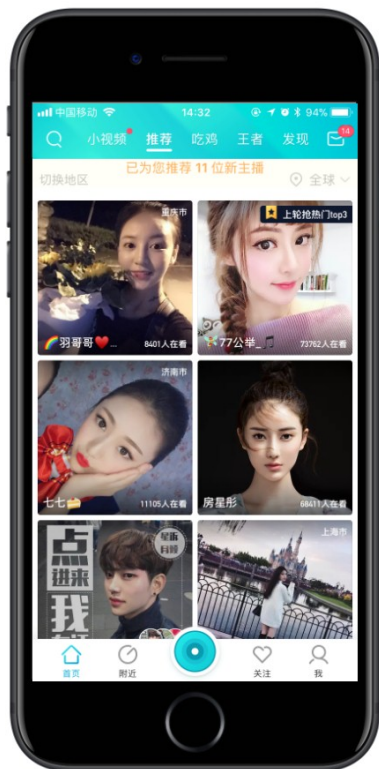
小程序播放器

-
- **特定类目**：如果您的小程序所属类目符合[类目要求](#)，可以使用 <live-player> 标签实现 iOS 和 Android 端的低延时直播，<live-player> 标签通过内置腾讯云 SDK 的精简版本实现，可以参考 [DOC](#) 对接低延时直播功能。
 - **其他类目**：如果是其他类目，则只能使用 <video> 标签 + HLS (m3u8) 协议实现高延时直播。

自由开播（UGC+OGC）

最近更新时间：2018-07-24 18:45:50

主播自由开播（UGC + OGC）解决方案，是指主播可以随时拿起手机开始直播，映客、花椒、斗鱼、Now 等直播平台都是采用这种直播解决方案。相比于手工生成一两条直播 URL 就能跑起来的单场次直播解决方案，自由开播方案需要您关注 **房间管理** 相关的逻辑，也就是维护一个所有用户可见的“直播间列表”。



映客直播

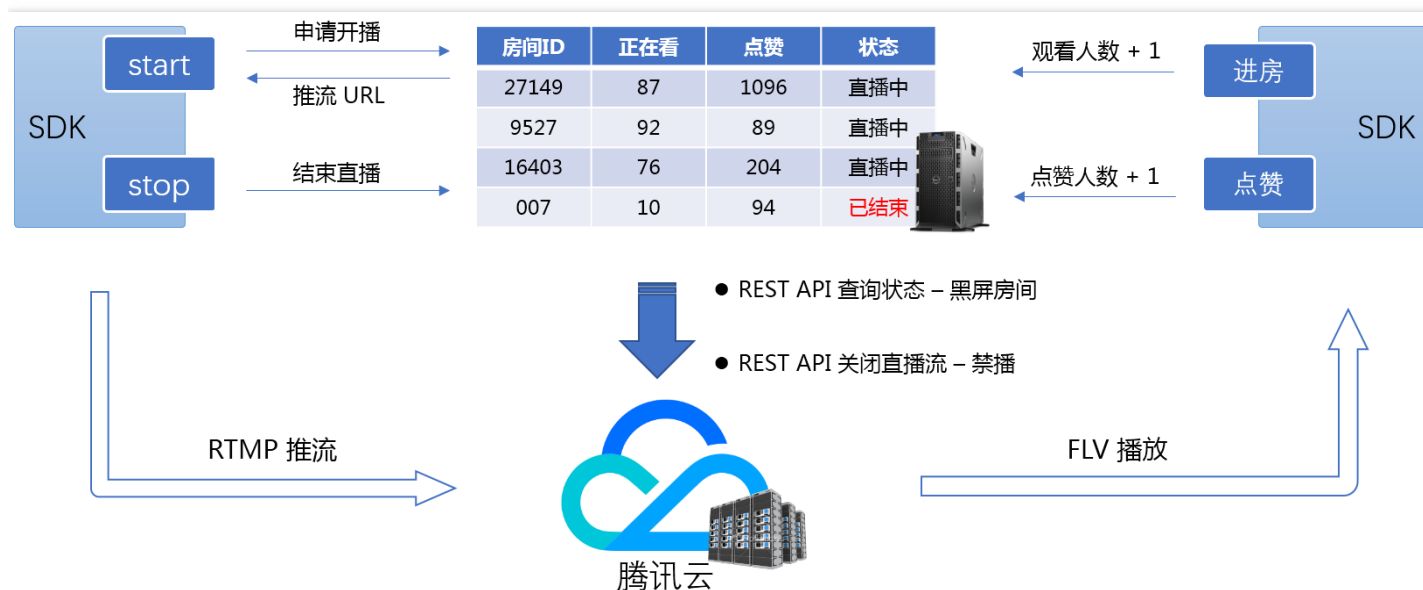


花椒直播



斗鱼直播

针对房间列表的管理和维护，无外乎增（ADD）、删（DELETE）、改（MODIFY）、查（QUERY）四个操作：



ADD：房间开播

当一个主播开播前需要先申请创建一个直播房间，这就等于是在您服务器的房间列表中增加一条新的数据。

- step1：主播请求开播(Client -> Server)**

Client 需要把主播的账号ID、直播间标题、直播封面URL、地理位置（非必须）等等信息统一提交给您的 Server。

- step2：Server预创建房间(Server -> Client)**

Server 要在直播间列表中添加一条记录，并将其状态设置为“等待开播（**unactive**）”，在 Server -> Client 的回包中需要返回推流 URL，这是主播开播所必须的。

- step3：主播开始推流（TXLivePusher）**

Client 在拿到推流 URL 后，就可以交给 SDK 开始推流，SDK 会将推流成功与否的情况通过 TXLivePushListener 回调通知给您的代码。

- step4：主播确认开播(Client -> Server)**

主播并不一定能够成功推流，比如：推流用的端口 1935 被所处网络的安全防火墙禁用了，或者是 App 刚刚安装，在看到摄像头权限申请提示时误点了拒绝。所以，step4 的目的就是在 Client 收到 SDK 的推流成功事件（ID: 1003）后，再通知后台把房间状态从“等待开播（**unactive**）”切换到“正在直播（**active**）”。

DELETE：关闭房间

直播结束后，Client 要通知后台把当前房间状态修改为“直播已结束（**close**）”，或者干脆将其从列表中删除。

- step1：主播停播(Client -> Server)**

主播停播时，Client 要通知服务器停播的直播流 ID，服务器可以把对应的房间状态改成“直播已结束（**close**）”，或直接将其从房间列表删除。

- **step2：解决黑屏房间(Server -> 腾讯云)**

如果主播的手机意外断网，或者 App 意外崩溃了，Client 会丧失通知服务器的机会，从而在房间列表中残留一些黑屏房间（主播已经没法推流了，但是没有人关闭这些房间，所以观众进入房间后看到的是黑屏）。

您的 Server 可以通过腾讯云 REST API ([Live_Channel_GetStatus](#)) 定时（推荐 10s 一次）地检查所有“正在直播（**active**）”状态中的房间是不是真的都是“正在推流”状态。如果某个房间在连续三次的查询结果中均为“离线”状态，Server 就可以判定其为“黑屏房间”并将其关闭了。

SDK 在网络短暂断开又恢复后会自动重连，重连期间您查询到的房间状态可能不准确，所以推荐以三次查询结果均为“离线”作为最终判定结论。

MODIFY：房间信息

很多场景下，我们都需要修改某一个房间的信息，比如：

- **加观众数(Client -> Server)**

当有新的观众加入时，意味着某个房间的观众数要 + 1，可以让 App 的观众端在进入房间时向 Server 发送一次累加请求。

- **加点赞数(Client -> Server)**

当有观众给主播点赞，意味着某个房间的点赞数要+1，可以让 App 在点赞按钮的响应函数中向 Server 发送一次累加请求。

补充：完整的点赞实现方案还要用聊天室的消息通道将点赞消息广播给所有的观众。

- **违规禁播(Server -> 腾讯云)**

当监管人员发现某一房间内容涉及违规行为时，需要对其禁播，意味着该房间的状态要改为“直播已结束（**close**）”，同时，禁播需要您的 Server 通过 REST API [开启关闭推流](#) 通知腾讯云立刻中断推流。

补充：腾讯视频云的鉴黄服务，是通过定时截图的方式，帮您甄别众多直播间中是否有疑似涉黄的直播流，并将嫌疑直播流的 ID 通过您指定的地址通知给您的后台 Server。这项服务目前还处于 beta 阶段，暂不支持自助开通，如有请通过 400 电话或者工单联系我们。

QUERY：房间列表

每一个打开App的观众，都会到直播后台查询一下当前的房间列表，所以直播后台要提供列表拉取的相关接口供 App 使用。

- **注意分页逻辑**

如果列表中房间数量比较多，比如100个以上，就推荐要加上分页逻辑了，分页逻辑对于减少服务器压力，提高列表展示速度方面非常有帮助。

- **拼装播放地址**

有了直播码（或者房间ID），播放地址就可以简单拼装出来，下图是用直播码 **8888_test_12345_test** 拼装出来的rtmp flv 和 hls 三种播

放地址，App拿到播放URL后就可以直接丢给腾讯云的RTMP SDK进行播放：

rtmp	rtmp://8888.liveplay.myqcloud.com/live/8888_test_12345_test
flv	http://8888.liveplay.myqcloud.com/live/8888_test_12345_test.flv
hls	http://8888.liveplay.myqcloud.com/live/8888_test_12345_test.m3u8

- **不要在 Client 拼装播放地址**

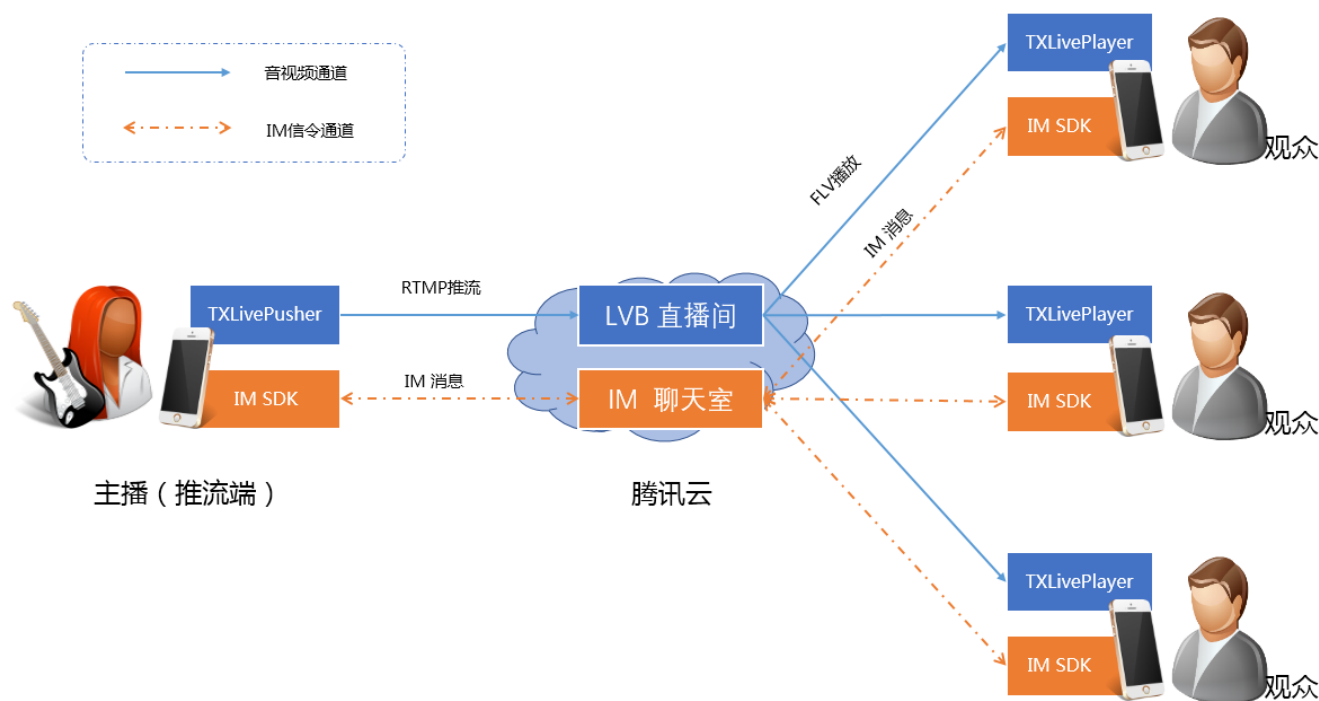
播放地址要由服务器下发，而不是 Client 拼装，这样能让您的系统更具灵活性。随着业务的发展，您可能会考虑在播放端增加播放防盗链，避免您的直播视频被盗用，而播放防盗链签名只有可能在服务器签发，故客户端拼装逻辑根本无法满足这个需求。

弹幕聊天室 (IM)

最近更新时间：2018-07-10 14:49:17

在线直播跟 IM 聊天室的关系是密切的，没有聊天室的互动消息，直播会变得乏味和无趣。本文主要介绍如何使用 **腾讯云通讯 (IM)** 服务构建一些简单的聊天室功能：

- (1) 群发弹幕、点赞消息。
- (2) 群发系统通知，比如“XXX已加入房间”或者“XXX已离开”等等。



1. 腾讯云 IM 服务

腾讯云 IM 的前身是 QQ 的即时通讯消息系统，我们将 QQ 的消息模块进行了抽离，变成了适合移动端接入的 IM SDK。将消息后台进行改造，使其脱离对 QQ 号码的绑定，构成了现在的 IM 云通讯后台。

您可以把 IM SDK 理解为一个没有用户交互界面的 QQ，把 IM SDK 集成到您的 APP 里，就相当于把一个 QQ 的消息内核集成在您的 APP 内部。

我们都知道，QQ 可以用来收发单聊和群聊的消息，但前提是您必须先登录才能使用。我们也都知道，登录 QQ 用的是 QQ 号和密码，登录 IM SDK 也是一样，只是肯定不能再使用 QQ 号和密码了，而是使用您指定的用户名 (userid) 和密码 (usersig)。

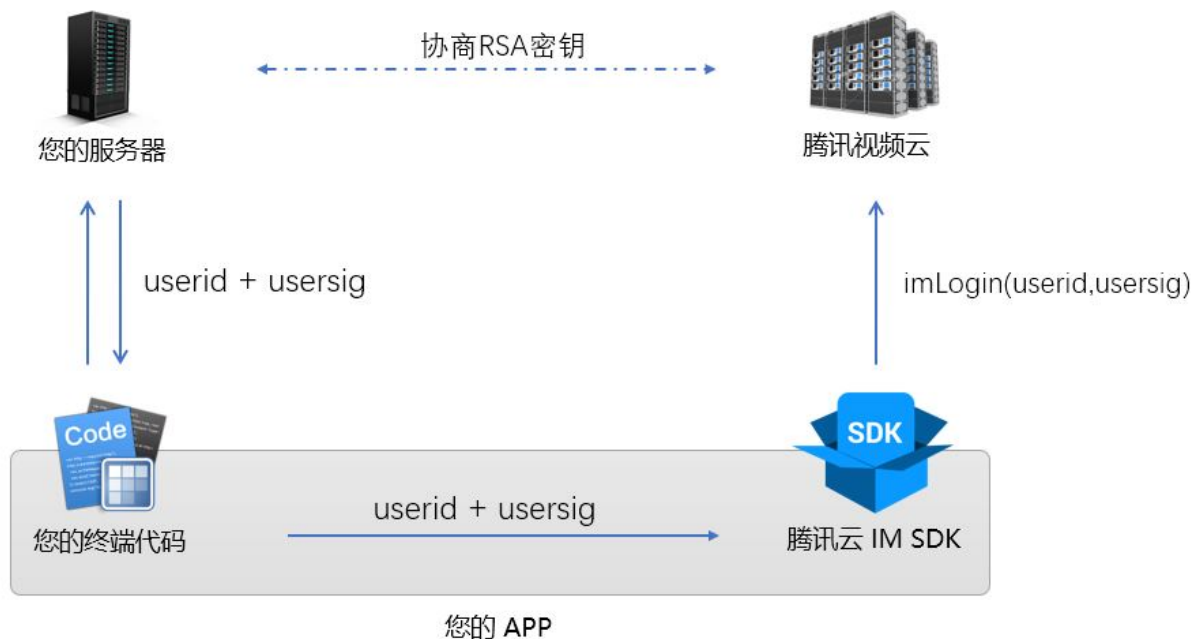
- **用户名 (userid)**

可以是您目前 APP 里的用户 ID，比如您有一个用户，他/她的账号 ID 是 27149，那么您就可以用 27149 作为登录 IM SDK 的 userid。

- **密码 (usersig)**

既然您指定了 27149 是您的用户，腾讯云如何才能确认该用户是您认可的合法用户呢？usersig 就是用于解决这个问题的，usersig 本质是对 userid、appid 等信息的非对称加密。

非对称加密用的加密密钥和解密密钥是不同的，您的服务器可以持有私钥并对 `userid` 和 `appid` 进行非对称加密，加密之后的结果就是 `usersig`；而腾讯云同步持有您的公钥，这样一来，腾讯云就可以确认 `usersig` 是否合法，从而可以确认它是否是由您的服务器签发的。



2. 服务开通&配置

进入云通讯[管理控制台](#)，如果还没有开通服务，直接点击 **开通云通讯** 按钮即可。新认证的腾讯云账号，云通讯的应用列表是空的，如下图：



点击 **创建应用接入** 按钮创建一个新的应用接入，即您要接入腾讯云 IM 通讯服务的 App 的名字，我们的测试应用名称叫做“小直播演示”，如下图所示：

创建新应用

创建方式

新建应用

应用名称

小直播演示

应用类型

视频

应用简介

演示云通讯接入

不超过300字

确定

取消

点击 **确定** 按钮，之后就可以在应用列表中看到刚刚添加的项目了，如下图所示：

应用列表				
创建应用接入 查看录像 下载云通信SDK				
SDKAPPID	应用名称	应用状态	创建时间	操作
1400069256	小直播演示	启用	2018-02-22 20:45:35	应用配置 统计分析
		启用	2018-01-13 22:36:39	应用配置 统计分析
		启用	2017-11-21 17:02:08	应用配置 统计分析

点击 **应用配置** 链接，会进入应用配置界面，再点击 **账号体系集成** 右侧的 **编辑** 按钮，按照下图所示进行配置即可（账号名称和管理员名称推荐用英文，账号名称随便填写，管理员工程在调用 IM 的 REST API 时可以用到）。

腾讯云

总览

云产品

常用服务

备案

直播 点播 对象存储 关系型数据库 云服务器 云通信 SSL证书管理 域名管理 微信小程序 服务商管理 +

云通信

应用列表

统计分析

应用名称

应用类型

应用简介

创建时间

上次修改时间

应用平台

编辑

所属平台

帐号体系集成

编辑

未配置

帐号体系集成

通过帐号登录集成，我们支持您创建的应用采用自有帐号，及QQ，微信等第三方开发

帐号名称

test

此项为必填项

集成模式

独立模式

托管模式

了解集成模式

帐号管理员

admin

什么是帐号管理员

+ 添加管理员111

保存

取消

点击 **保存** 按钮，页面会自动刷新，之后就可以看到 **下载公私钥** 的按钮了。



点击 **下载公私钥** 按钮，会得到一个叫做 **keys.zip** 的压缩包，里面有一个 private_key 和一个 public_key，**private_key** 就是用来签发 UserSig 的私钥了。

操作系统 (C:) > 用户 > rexchang > 下载 > keys.zip						搜索"keys.zip"	
名称	类型	压缩大小	密码保护	大小	比率		
private_key	文件	1 KB	否	1 KB	18%		
public_key	文件	1 KB	否	1 KB	15%		

可以在 **开发辅助工具** 中测试一下 `private_key` 是否能正常进行签名。



3. Server 接入指引

3.1 派发UserSig

后台需要做的事情非常简单，就是向 Client 派发 UserSig，我们在上一步中经过简单的几次鼠标点击，已经获得了用于签发 UserSig 的公私钥。

接下来，你可以通过阅读 [DOC](#) 了解各个语言版本（Java、PHP、C++）的 UserSig 生成代码，之后可以将其集成到您的后台系统中。

推荐的做法是将其集成到登录环节中，也就是在用户登录时，您的后台服务器除了返回之前应该返回的信息，还可以将 UserSig 也一并返回给您的 App。

3.2 调用 REST API

您还可以通过调用 [REST API](#) 来进行更多的二次开发，比如：

API	中文含义	用途
v4/group_open_http_svc/create_group	创建群组	一次直播对应的聊天室可以由主播在 Client 端通过 IM SDK 来创建，也可以由您的 Server 通过此 API 进行创建
v4/group_open_http_svc/send_group_msg	发送消息	可以将消息用这个 API 推送给群组里的每一个成员
v4/openim_dirty_words/add	添加脏字	用于敏感字过滤。

4. Client 接入指引

4.1 下载 SDK

到云通讯官网可以下载 [IM SDK](#) 的最新版本，推荐 v3 版本，接入难度相比于 v2 更低。

4.2 IM登录(imLogin)

您的 Server 签发的 userSig，可以在 App 连接到服务器时返回给 App，App 在拿到 userid + usersig 后，就能登录到腾讯云通讯服务收发消息了（这就好比在输入正确的 QQ 号和密码之后，就可以登录到 QQ 后台收发消息了），代码示例如下：

```
// iOS 端示例代码: imLogin
//
#import "ImSDK/ImSDK.h"
TIMLoginParam *param = [[TIMLoginParam alloc] init];
// identifier为用户名, userSig 为用户登录凭证, 服务器计算
param.identifier = _config.userID;
param.userSig = _config.userSig;
param.appidAt3rd = [NSString stringWithFormat:@"%d", _config.appID];

[[TIMManager sharedInstance] login:param succ:^(
//登录成功
) fail:^(int code, NSString *msg) {
//登录失败
}];

// android 端示例代码: imLogin
//
TIMUserConfig timUserConfig = new TIMUserConfig();
TIMManager.getInstance().login(identifier, userSig, timUserConfig, new TIMCallBack() {
@Override
public void onError(int code, String desc) {
//错误码code和错误描述desc, 可用于定位请求失败原因
//错误码code列表请参见错误码表
Log.d(tag, "login failed. code: " + code + " errmsg: " + desc);
}

@Override
```

```
public void onSuccess() {
    Log.d(tag, "login succ");
}
});
```

4.3 加入聊天室

IM SDK 只有在加入特定的聊天室之后才能收发群消息，加入聊天室的示例代码如下：

```
// iOS 端示例代码: 加入聊天室
//
[[TIMGroupManager sharedInstance] joinGroup:@"TGID1JYSZAEQ" msg:@"Apply Join Group" succ:^(){
    NSLog(@"Join Succ");
}fail:^(int code, NSString * err) {
    NSLog(@"code=%d, err=%@", code, err);
}];

// Android 端示例代码: 加入聊天室
//
TIMGroupManager.getInstance().applyJoinGroup(groupId, "", new TIMCallBack() {
    @Override
    public void onError(int i, String s) {
        Log.d(TAG, "加入群 失败");
    }

    @Override
    public void onSuccess() {
        Log.d(TAG, "加入群"+groupId+"成功");
    }
});
```

4.4 发送群聊消息

```
// iOS 端示例代码: 发送群聊消息
//
TIMTextElem * text_elem = [[TIMTextElem alloc] init];
[text_elem setText:@"this is a text message"];
TIMMessage * msg = [[TIMMessage alloc] init];
[msg addElem:text_elem];
[conversation sendMessage:msg succ:^(){
    NSLog(@"SendMsg Succ");
}fail:^(int code, NSString * err) {
    NSLog(@"SendMsg Failed:%d->%@", code, err);
}];

// Android 端示例代码: 发送群聊消息
//
TIMTextElem textElem = new TIMTextElem();
textElem.setText(msgText);
TIMMessage message = new TIMMessage();
message.addElement(textElem);
TIMConversation conversation = TIMManager.getInstance().getConversation(TIMConversationType.Group, groupId);
conversation.sendMessage(message, new TIMValueCallBack<TIMMessage>(){
```

```
@Override
```

```
public void onError(int i, String s) {  
    Log.d(TAG, "发送群消息失败");  
}
```

```
@Override
```

```
public void onSuccess(TIMMessage timMessage) {  
    Log.d(TAG, "发送群消息成功");  
}  
});
```

4.5 更多 API

这里仅摘录了最常用的几个 API, 更多 IM SDK 的接口可以在腾讯云通讯的[文档区](#)查找到。

录制回看 (VOD)

最近更新时间：2018-03-23 23:16:37

功能介绍

所谓录制回看，是指您可以把用户整个直播过程录制下来，然后作为点播视频用于回看。

在APP上线的初期阶段，由于主播数量比较少，所以在直播列表中加入录制回看，能够在一定程度上丰富APP在观众端的信息量。

即使到APP成长起来主播数量形成规模以后，好的直播内容的沉淀依然是必不可少的一个部分，每个主播的个人介绍里除了有名字、照片和个人信息，历史直播的视频回看更是不可或缺的一个重要组成部分。



开启录制

录制回看功能依托于腾讯云的**点播服务**支撑，如果您想要对接这个功能，首先需要在腾讯云的管理控制台**开通点播服务**。服务开通之后，新录制的文件就可以在点播控制台的**视频管理**里找到它们。

那么怎么开启录制呢？这里有两个办法：

1. 全局开启录制

即指定所有直播的视频流全部开启或者关闭录制，在**直播管理控制台 >> 接入管理 >> 直播码接入 >> 接入配置**中可以对其进行设置，见下图：

直播录制

直播录制为按月计费功能，开启即收费。收费标准：每录制频道30元/月。频道数取月并发表录制频道峰值。 [查看详细](#)

直播录制



录制文件类型



FLV



MP4



HLS

保存

取消

注意：全局录制的分片时长默认为30分钟，如果您需要配置全局录制的分片时长，可以提工单申请。

2. 指定房间录制

在所有直播的视频流全部关闭录制的情况下，您依然可以对个别重要的视频流开启录制，操作方法是在推流URL后拼接 **&record=mp4** 或者 **&record=hls** 或者 **&record=mp4|hls**，例如：

```
rtmp://2121.livepush.myqcloud.com/live/2121_15919131751?txSecret=aaa&txTime=bbb&record=mp4&record_interval=5400
```

这里有几点要特别说明：

- 目前录制功能支持的视频封装格式有 mp4、hls 和 flv 三种，关于点播格式的差异，您可以参考这里 [DOC](#)。
- record=mp4|hls|flv 分隔符格式用于指定同时录制一种以上的视频格式（只有 MP4 和 HLS 支持手机浏览器播放）。
- mp4 格式的视频是 不支持 直播中途切换分辨率或做横竖屏切换的。
- 如果您指定的录制格式是 flv 或 mp4，可以通过 record_interval 参数用于指定单个录制分片的时长，单位是秒，最长支持120分钟（也就是7200），不指定的话默认值是30分钟（也就是1800）。
- hls（m3u8）文件本身就是小分片机制，所以无所谓切断问题，只要直播过程中不断流，您只会拿到一个m3u8文件。但如果直播期间推流出现中断，录制过程依然会出现分段问题（也就是会得到多个m3u8），其中一个常见的问题就是 App 切后台，推荐采用后台推流解决方案进行缓解。
- 如果您想只录制音频，可以通过添加参数 record_type=audio 来指定。

获取文件

一个新的录制视频文件生成后，会相应的生成一个观看地址，您可以按照自己的业务需求对其进行处理。在小直播中，我们直接将录制的文件URL和房间列表拼在了一起，以弥补在线主播不足的窘境。

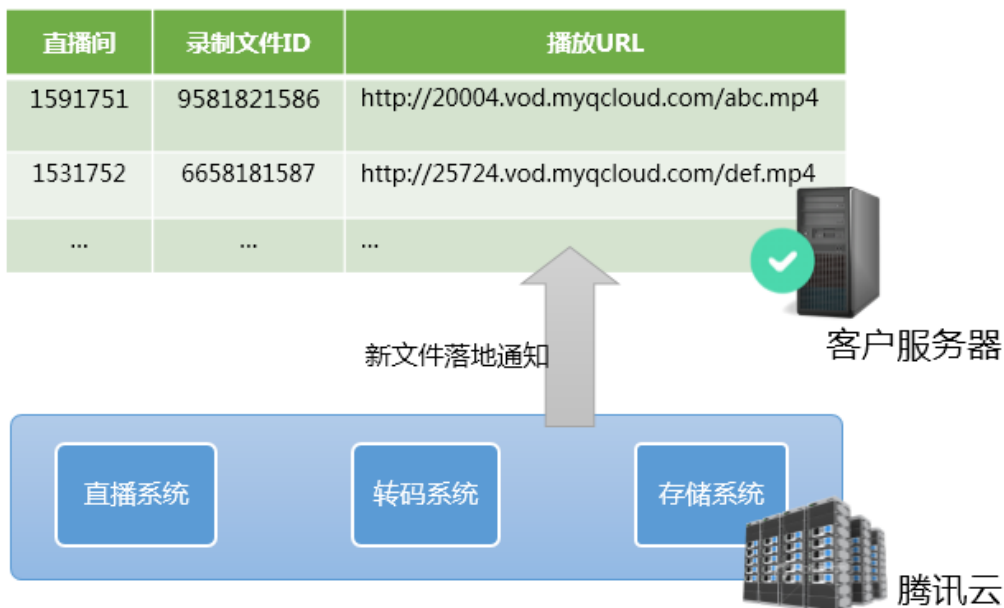
但您可以结合自己的业务场景实现很多的扩展功能，比如：您可以将其追加到主播的资料信息里，作为该主播曾经直播的节目而存在；或者将其放入回放列表中，经过专门的人工筛选，将优质的视频推荐给您的App用户。

那么怎样才能拿到文件的地址呢？有如下两种解决方案：

1. 被动监听通知

您可以使用腾讯云的[事件通知服务](#)：您的服务器注册一个自己的回调URL给腾讯云，腾讯云会在一个新的录制文件生成时通过这个URL通知给您。

录制文件管理表



如下是一个典型的通知消息，它的含义是：一个id为9192487266581821586的新的flv录制文件已经生成，播放地址为：`http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv`。

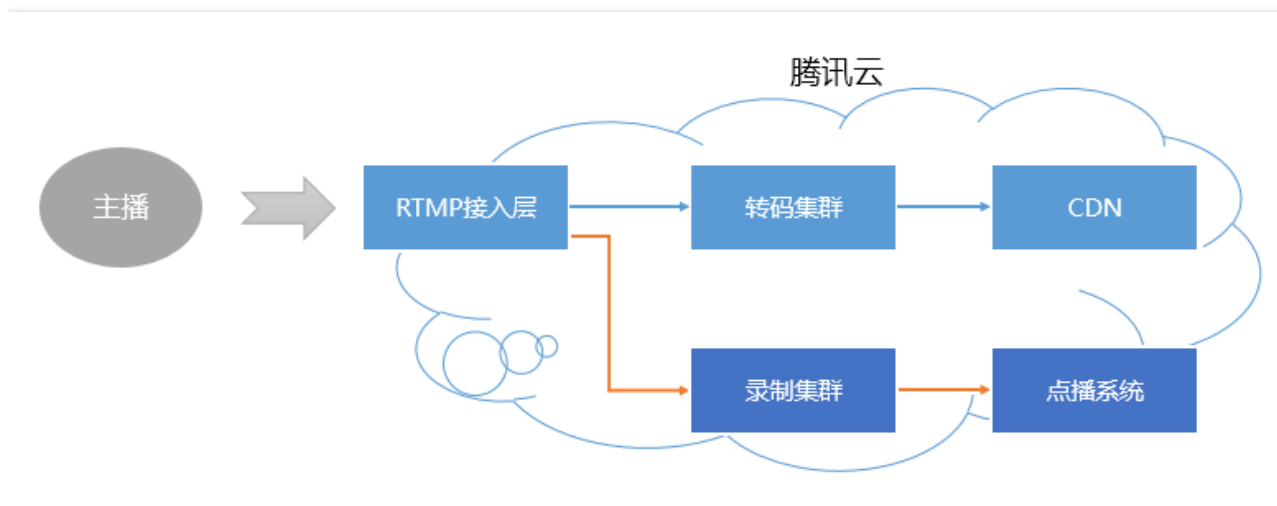
```
{
  "channel_id": "2121_15919131751",
  "end_time": 1473125627,
  "event_type": 100,
  "file_format": "flv",
  "file_id": "9192487266581821586",
  "file_size": 9749353,
  "sign": "fef79a097458ed80b5f5574cbc13e1fd",
  "start_time": 1473135647,
  "stream_id": "2121_15919131751",
  "t": 1473126233,
  "video_id": "200025724_ac92b781a22c4a3e937c9e61c2624af7",
  "video_url": "http://200025724.vod.myqcloud.com/200025724_ac92b781a22c4a3e937c9e61c2624af7.f0.flv"
}
```

2. 主动查询文件

您可以通过腾讯云的文件查询接口（[Live_Tape_GetFilelist](#)）定时地查看是否有新的录制文件生成，不过这种方案在要查询的频道数特别多的时候，响应速度不理想，同时调用频率也不能太快（仅对刚结束的频道进行调用为宜），这种方案的实时性和可靠性不高，并不推荐频繁使用。

常见问题

1. 直播录制的原理是什么？



对于一条直播流，一旦开启录制，音视频数据就会被旁路到录制系统。主播的手机推上来的每一帧数据，都会被录制系统追加写入到录制文件中。

一旦直播流中断，接入层会立刻通知录制服务器将正在写入的文件落地，将其转存到点播系统中，并为其生成索引，这样您在点播系统中就会看到这个新生成的录制文件了。同时，如果您配置了录制事件通知，录制系统会将该文件的 **索引ID** 和 **在线播放地址** 等信息通知给您之前配置的服务器上。

但是，如果一个文件过大，在云端的转出和处理过程中就很容易出错，所以为了确保成功率，我们的单个录制文件最长不会超过90分钟，您可以通过 `record_interval` 参数指定更短的分片。

2. 一次直播会有几个录制文件？

- 如果一次直播过程非常短暂，比如只有不到 1 秒钟时间，那么可能是没有录制文件的。
- 如果一次直播时间不算长（小于 `record_interval`），且中途没有推流中断的事情发生，那么通常只有一个文件。
- 如果一次直播时间很长（超过 `record_interval`），那么会按照 `record_interval` 指定的时间长度进行分片，分片的原因是避免过长的文件在分布式系统中流转的时间不确定性。
- 如果一次直播过程中发生推流中断（之后 SDK 会尝试重新推流），那么每次中断均会产生一个新的分片。

3. 如何知道哪些文件属于某一次直播？

准确来说，作为 PAAS 的腾讯云并不清楚您的一次直播是怎么定义的，如果您的一次直播持续了20分钟，但中间有一次因为网络切换导致的断流，以及一次手动的停止和重启，那么这算是一次直播还是三次呢？

对于普通的移动直播场景，我们一般定义如下的界面之间的这段时间为一次直播：



所以来自 APP 客户端的时间信息很重要，如果您希望定义这段时间内的录制文件都属于这次直播，那么只需要用直播码和时间信息检索收到的录制通知即可（每一条录制通知事件都会携带stream_id、start_time 和 end_time 等信息）。

4. 如何把碎片拼接起来？

目前腾讯云支持使用云端 API 接口拼接视频分片，API 详细用法可以参考 [视频拼接](#)。

直播连麦 (LiveRoom)

最近更新时间：2018-08-06 10:30:40

功能介绍

直播+连麦 是在 **秀场直播** 和 **在线教育** 场景中经常使用的直播模式，它既能支持高并发和低成本的在线直播，又能通过连麦实现主播和观众之间的视频通话互动，具有极强的场景适用性。



LiveRoom

如果您只需要做一个简单的直播方案，也就是：主播推流 + 观众观看 + 文字互动，参考[单场次直播](#)，或者[自由开播](#)就可以解决问题。

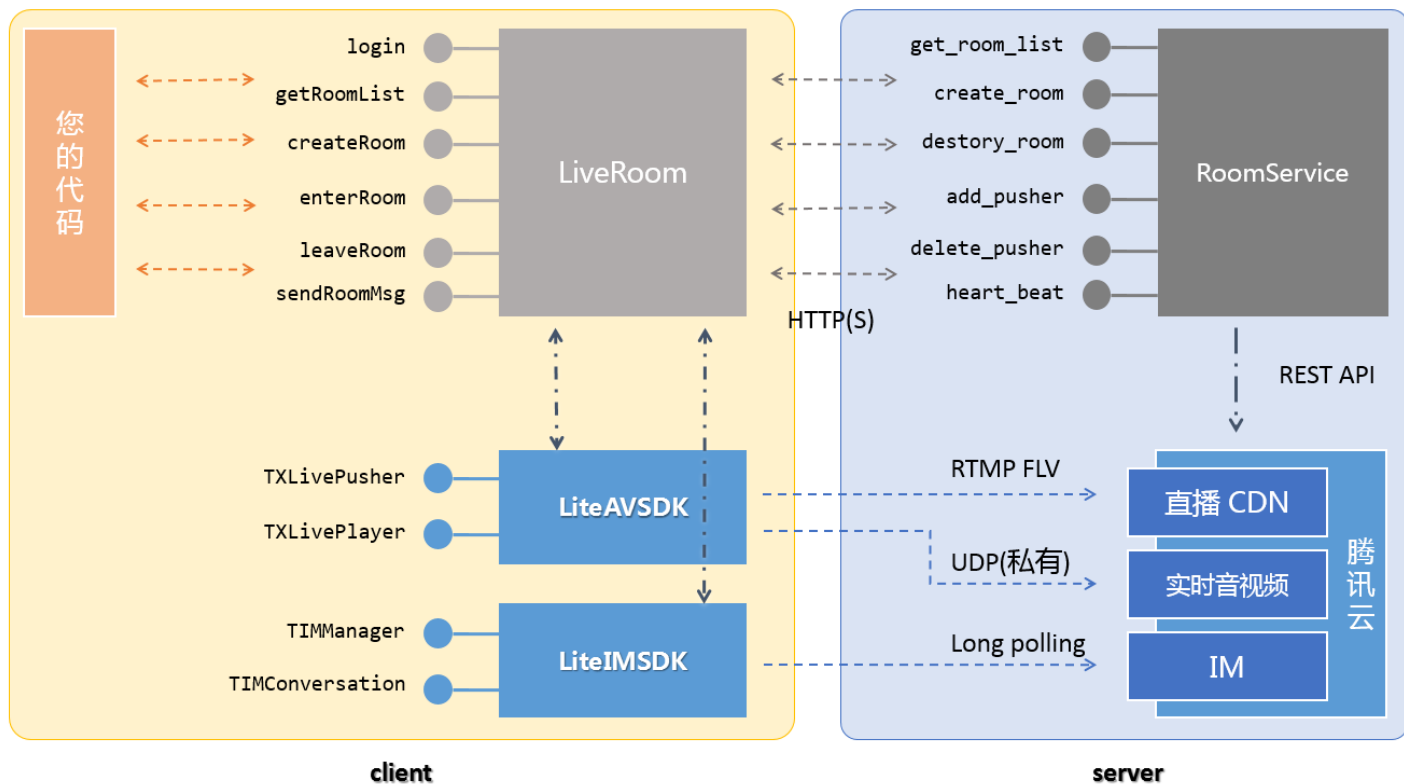
但是如果您要上连麦功能，单纯依靠 TXLivePusher 和 TXLivePlayer 两个组件进行组合，那难度可就高了，为了降低这里的实现难度，我们推出了 **LiveRoom** 组件，它分成 Client 和 Server 两个部分：

• 终端部分 (Client)

LiveRoom 组件的终端部分是对腾讯视频云 LiteAVSDK（主要用于音视频，包括 TXLivePusher、TXLivePlayer 等接口）和 LiteIMSDK（主要用于收发消息，包括 TIMManager 和 TIMConversation 等接口）的封装。直接使用 LiteAVSDK 和 LiteIMSDK 实现直播 + 连麦是非常耗时耗力的，但通过 LiveRoom 组件，您可以直接调用 createRoom，enterRoom 和 leaveRoom 等接口就可以完成您想要的直播 + 连麦功能。

• 后台部分 (Server)

RoomService 是 LiveRoom 对应的后台组件，其职责有两个：一是房间管理（直播间的增、删、改、查）和成员管理（尤其是要维护好房间里有几个人正在跟主播连麦）；二是对腾讯云直播服务、实时音视频服务以及 IM 云通讯服务的控制（主要通过腾讯云的后台 REST API 进行调用）。



终端对接

step1: 下载 SDK 开发包

平台	编程语言	SDK 下载	API 文档
iOS	Objective-C	DOWNLOAD	API 文档
Android	java	DOWNLOAD	API 文档
IE浏览器	javascript	DOWNLOAD	API 文档
PC (C++)	C++	DOWNLOAD	API 文档

- LiteAV SDK 位于 ZIP 包解压后的 SDK 文件夹中，用于实现音视频相关功能
- LiteIM SDK 位于 ZIP 包解压后的 SDK 文件夹中，用于实现 IM 通讯相关功能
- LiveRoom 位于 ZIP 包解压后的 Demo \ liveroom 文件夹中，代码开源，方便您进行调试和定制。

step2: 登录 (login)

LiveRoom 单靠一个终端的组件无法独自运行，它依赖一个后台服务为其实现房间管理和状态协调，这个后台服务我们称之为**房间服务**（RoomService）。而使用这个房间服务，LiveRoom 就需要先进行**登录**（login）。

阅读 [RoomService](#) 可以了解 login 函数的几个参数应该如何填写。

step3: 获取房间列表（getRoomList）

不管是主播还是观众，不管是老师还是学生，都需要有一个房间列表，调用 LiveRoom 的 **getRoomList** 接口可以获取到该列表。列表中每一个房间都有其对应的 roomInfo，是在 createRoom 时传入的，推荐您将 roomInfo 定义为 json 格式，这样可以有很强的扩展性。

如果您希望使用自己的房间列表，这一步可以省略，但是需要您在 step 4 中自行指定 roomId，且全局不能有重复。

step4: 主播开播（createRoom）

主播要开播，需要先调用 LiveRoom 的 **startLocalPreview** 接口开启本地摄像头预览，该函数需要传入一个 view 对象，该对象用于显示摄像头的视频影像。这期间 LiveRoom 会申请摄像头使用权限，同时，主播也可以对着摄像头调整一下美颜和美白的具体效果。

之后，通过调用 **createRoom** 接口，LiveRoom 会在后台的房间列表中新建一个直播间，同时主播端会进入推流模式。

参数roomId

如果您在调用 createRoom 时不填写 roomId，后台会为您分配一个 roomId，并通过 createRoom 的回调接口返回给您。如果您希望自己管理房间列表，roomId 可以由您的服务器分配，那么只需要在调用 createRoom 时填写您后台分配的 roomId 即可。

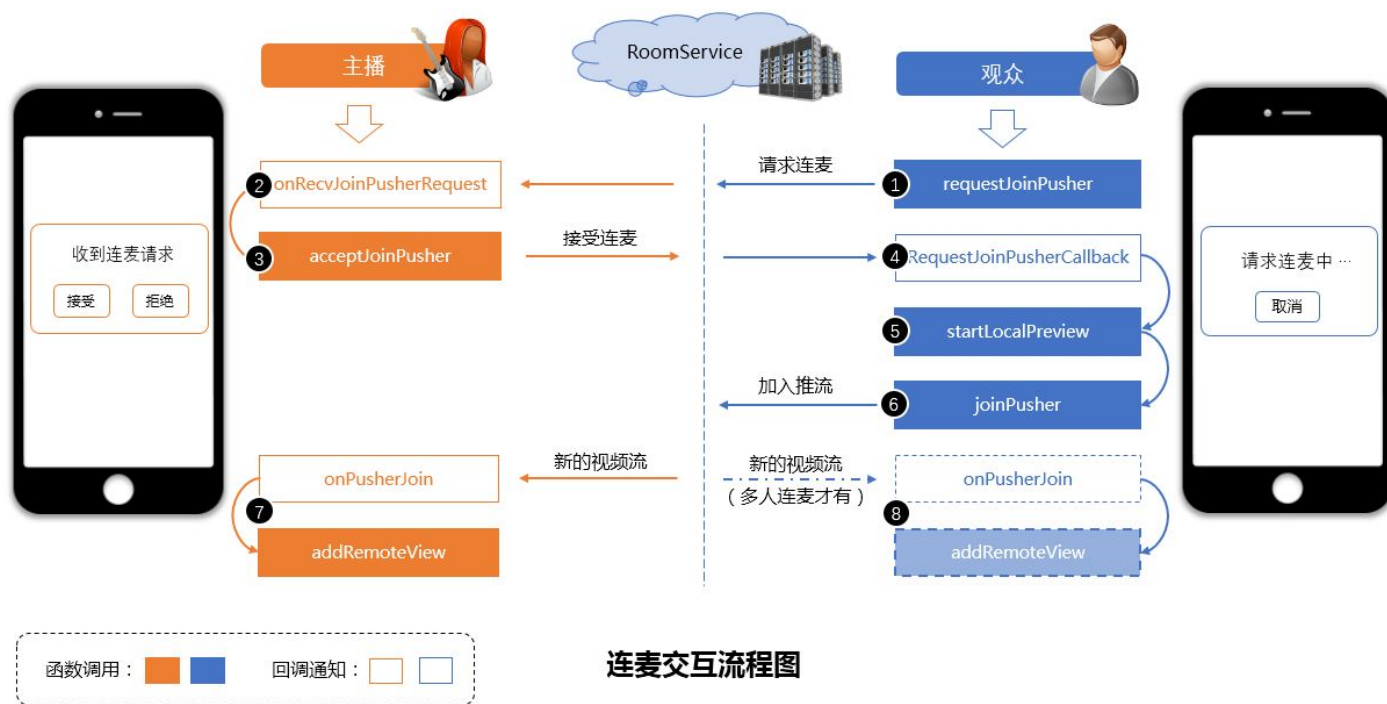
step5: 观看直播（enterRoom）

观众通过 LiveRoom 的 **enterRoom** 接口可以进入直播间观看视频直播，enterRoom 函数需要传入一个 view 对象，用于显示直播流的视频影像。

另外，进入房间后，调用 LiveRoom 的 **getAudienceList** 接口可以获取观众列表，这里的列表不是全量数据，如果少于30 个人就全部返回，如果多于 30 个人，就只返回新进入的 30 个人。（出于性能方面的考虑，而且 UI 界面上最多能也就能放下 10 个头像。）

step6: 连麦（joinPusher）

连麦是一个需要主播和观众共同参与的过程，可以参考如下流程图：



连麦交互流程图

- 第一步（观众）：首先通过 `requestJoinPusher` 发起连麦请求。
- 第二步（主播）：会收到 `onRecvJoinPusherRequest` 回调通知，之后可以展示一个 UI 提示，询问主播要不要接受连麦。
- 第三步（主播）：可以选择 `acceptJoinPusher` 接受连麦，也可以选择 `rejectJoinPusher` 拒绝连麦。
- 第四步（观众）：通过 `RequestJoinPusherCallback` 可以了解到连麦请求是否被接受，如果被接受。
- 第五步（观众）：如果被连麦请求被接受，需要调用 `startLocalPreview` 开启本地摄像头，`startLocalPreview` 需要一个 view 对象用于显示摄像头的影像，此时 SDK 会申请本地摄像头和麦克风的权限。
- 第六步（观众）：之后，通过调用 `joinPusher` 函数可以开始进入连麦状态（观众端开始推流）。
- 第七步（主播）：一旦观众开始进入连麦状态，主播端就会收到 `onPusherJoin` 通知，该通知会将连麦者的信息封装在一个叫做 `pusherInfo` 的对象里通知出来。接下来主播会通过 `addRemoteView` 函数把连麦者的远程影像显示出来。
- 第八步（观众）：如果房间里已经有一个（以上）的连麦者，那么观众端也会收到 `onPusherJoin` 通知，也可以通过 `addRemoteView` 把其它连麦者的影像显示出来。

如果您要实现的是 1v1 连麦，那么上述流程中的第 8 步是可以忽略的。

step7: 弹幕消息 (sendMsg)

LiveRoom 自带了消息发送接口，可以通过 `sendRoomTextMsg` 函数发送普通的文本消息（用来弹幕），也可以通过 `sendRoomCustomMsg` 发送自定义消息（用于点赞，送花等等）。

通过 `RoomListenerCallback` 里的 `onRecvRoomTextMsg` 和 `onRecvRoomCustomMsg` 可以收取聊天室里别人发来的文本消息和自定义消息。

ATTENTION

腾讯云 IM 每秒钟最多可以收取 40 条以上的消息，如果您要把所有这些消息都按照接收频率刷新到屏幕 UI 上，那您的直播体验一定是非常卡顿的，这里一定要注意刷新频率控制。

有太多的客户在测试期间顺顺利利，APP一上线就卡的不行，都是这个原因导致的。

后台对接

LiveRoom 为什么需要 login ？

LiveRoom 单靠一个终端的组件无法独自运行，它依赖一个后台服务为其实现房间管理和状态协调，这个后台服务我们称之为**房间服务**（RoomService）。而要使用这个房间服务，LiveRoom 就需要先进行**登录**（login）。

login 有很多参数需要填写，我应当如何填写这些参数呢？

如下表格中列举了三种填写方案，每种方案都有其适用场景：方案一适合调试；方案二适合快速上线；方案三适合自行定制；

参数名	方案一（纯测试方案）	方案二（腾讯云RoomService）	方案三（自建RoomService）
serverDomain	使用腾讯云 RoomService https://room.qcloud.com/weapp/live_room	使用腾讯云 RoomService https://room.qcloud.com/weapp/live_room，需要提前配置	自行部署 RoomService https://[yourcompany]/weapp/live_room
sdkAppID	通过测试地址获取 https://room.qcloud.com/weapp/utls/get_login_info_debug	自行填写，如何获取？	自行填写，如何获取？
accType	通过测试地址获取 https://room.qcloud.com/weapp/utls/get_login_info_debug	自行填写，如何获取？	自行填写，如何获取？
userID	通过测试地址获取 https://room.qcloud.com/weapp/utls/get_login_info_debug	您来指定，比如 9527	您来指定，比如 9527
userSig	通过测试地址获取 https://room.qcloud.com/weapp/utls/get_login_info_debug	您的服务器生成，如何生成？	您的服务器生成，如何生成？
账号归属	账号为腾讯云测试账号	账号为您名下的账号	账号为您名下的账号
账号限制	每天 10：00 - 22：00 可用	无限制，但不支持定制	无限制，且您可以进行定制
适用场景	前期终端研发调试使用	产品上线初期阶段	产品进入上升期

方案一：纯调试方案

该方案使用腾讯云为方便客户调试而统一提供的测试账号，仅适合调试期间使用，每天 10:00 - 22:00 这个时间段可用。

step1. 配置RoomService

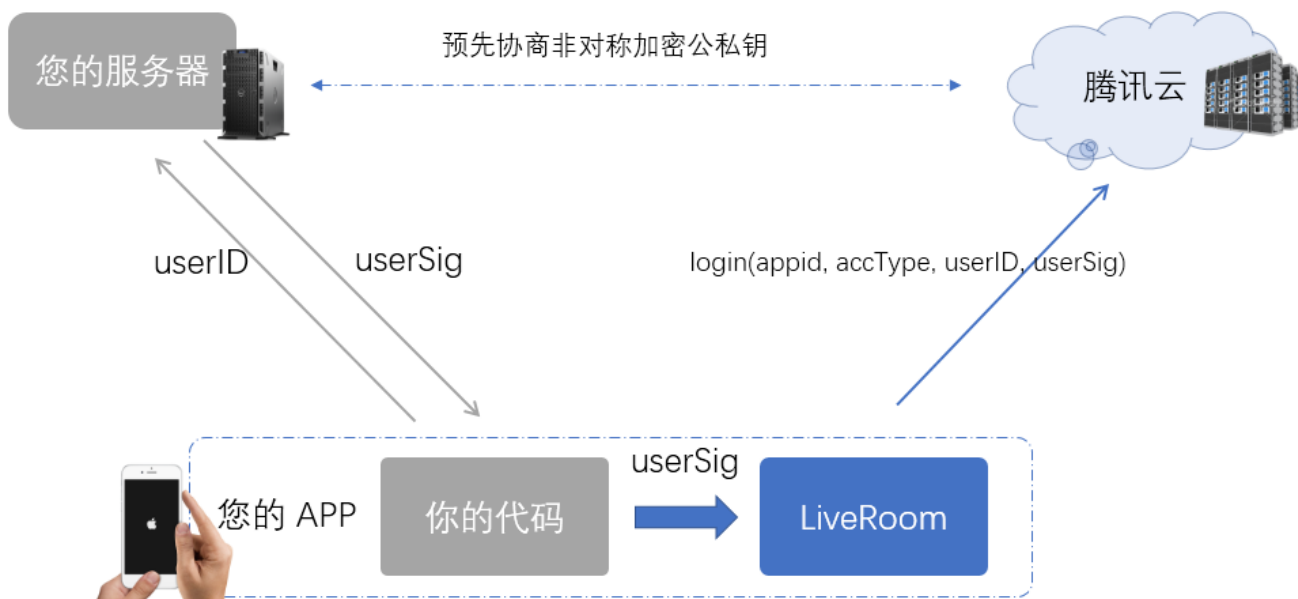
serverDomain 填写 `https://room.qcloud.com/weapp/live_room` 即可。

step2. 获得 login 所需参数

通过测试地址 (`https://room.qcloud.com/weapp/utis/get_login_info_debug`) 获取相关参数

方案二：使用腾讯云RoomService

该方案使用您自己的腾讯云账号 + 腾讯云自行部署的 RoomService 服务，所以在使用前，您需要先对 RoomService 进行配置。



step1. 配置RoomService

腾讯云 RoomService 地址为

`https://room.qcloud.com/weapp/live_room`

点击 [RoomTool.zip](#) 下载腾讯云 RoomService 后台配置工具，这是一个基于 Node.js 的配置工具，需要您在使用前 [安装](#) Node.js。配置工具压缩包中包含的 pdf 和 PPT 有详细的配置说明，这里仅简要概括一下各个配置项的含义和作用。

配置项	作用	获取方案
直播 (live) appId	腾讯云直播服务基于 appId 区分客户身份	DOC
直播 (live) APIKey	腾讯云直播服务的后台 REST API，采用 APIKey 进行安全保护	DOC
云通讯 (IM) sdkAppID	腾讯云通讯服务用 sdkAppID 区分 IM 客户身份	DOC
云通讯 (IM) accountType	曾用于区分 APP 类型，现仅出于兼容性原因而保留	DOC

配置项	作用	获取方案
云通讯 (IM) administrator	RoomService 使用了 IM REST API 发送房间里的系统消息，而 IM REST API 接口需要您填写管理员名称。	DOC
云通讯 (IM) privateKey	RoomService 使用 privateKey 用于签发管理员 (administrator) 的 usersig，进而能够调用 IM REST API 发送房间里的系统消息。	DOC
云通讯 (IM) publicKey	RoomService 使用 publicKey 用于确认终端用户的登录身份。	DOC

step2. 获得 login 所需参数

RoomService 需要终端调用 `login(serverDomain, sdkAppID, accType, userID, userSig)` 登录成功后才能使用，其中前四个信息都可以在客户端写死，但是 UserSig 必须由您的后台服务器签发，因为让 Client 计算 UserSig 就需要将签名私钥写在终端的代码里，这会有私钥被 hack 窃取的安全风险。

RoomService 所使用的 UserSig 的签发同 IM 云通讯服务是一样的，所以同一个 UserSig 既可以用来登录 IM，又可以用来登录 RoomService，您可以参考文档 [派发UserSig](#) 进行接入。

方案三：自建RoomService后台

该方案使用您自己的腾讯云账号 + 您自己部署的房间服务，所以您可以对内部的逻辑进行修改和定制。

step1. 下载源码 & 修改配置 & 部署

在 [CODE](#) 下载 RoomService 后台源码，源码包分成三个目录，其中 live_room 下的源码是您需要关注的。

下载到源码后，解压并找到 live_room 文件夹下面的 config.js 文件，这里有几个配置项需要修改。配置项跟方案一中的基本类似，可以参考方案一中的 step1 进行配置。区别在于，在方案二中您不需要用 RoomService 配置工具，而是直接修改本地源码即可。

```

16
17 /**
18  * 需要开通云直播服务
19  * 参考指引 @https://cloud.tencent.com/document/product/454/7953#1.-.E8.A7.86.E9.A2.91.E7.9B.B4.
20  * 有介绍bizid 和 pushSecretKey的获取方法。
21  */
22 live: {
23   /**
24    * 云直播 appID: 和 APIKey 主要用于腾讯云直播common cgi请求。appid 用于表示您是哪个客户，API
25    * 后台用他们来校验common cgi调用的合法性
26    */
27   /**
28    appID: 3,
29   */
30   /**
31    * 云直播 bizid: 和pushSecretKey 主要用于推流地址的生成。填写错误，会导致推流地址不合法，推流
32    */
33   bizid: ,

```

配置项	作用	获取方案
-----	----	------

配置项	作用	获取方案
直播 (live) pushSecretKey	用于计算推流 URL 的防盗链签名，必须要配置	DOC
直播 (live) APIKey	腾讯云直播服务的后台 REST API，采用 APIKey 进行安全保护	DOC

之后，您可以将其部署到您自己的后台服务器上，并将服务器的外网 URL 告知您的终端研发工程师，因为他/她在调用终端的 [login](#) 函数时需要指定 RoomService 的后台地址，例如：

```
https://[www.yourcompany.com]/weapp/live_room
```

step2. 获得 login 所需参数

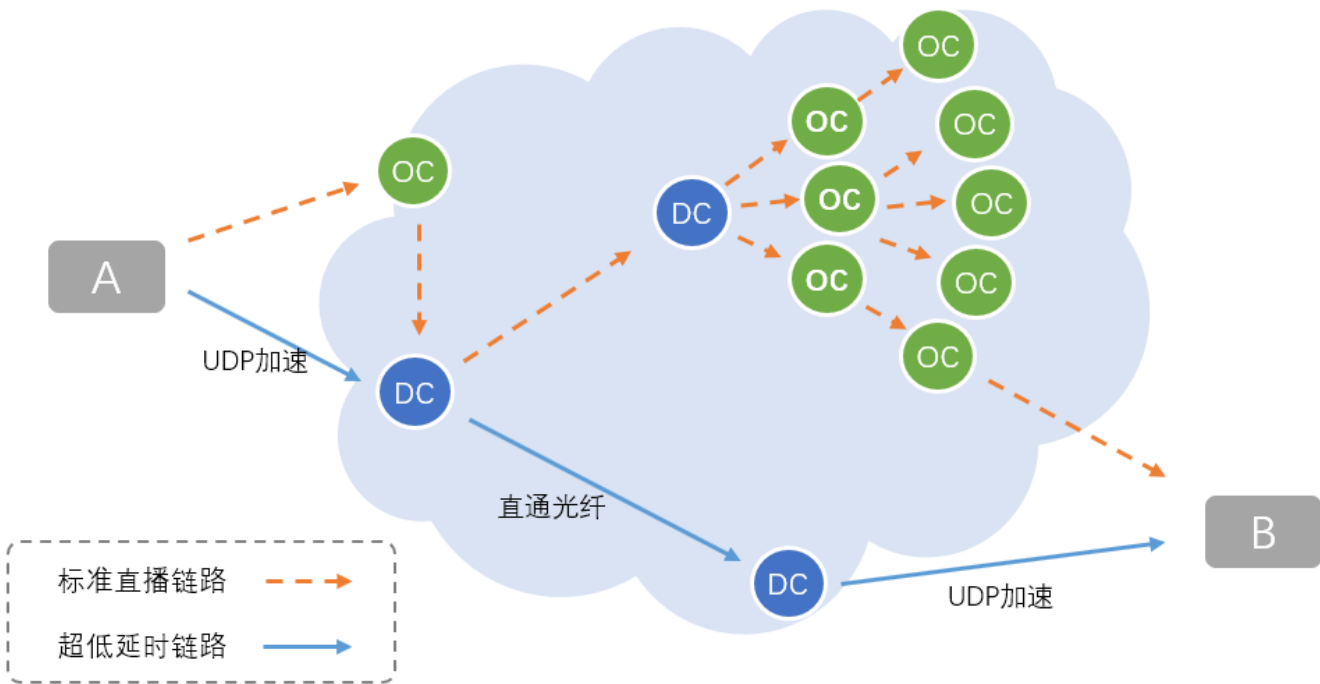
第一步完成之后，相当于您已经拥有了一个自己的 RoomService，但这个自己的 RoomService 还是基于 sdkAppID, accType, userID, userSig 这四个信息进行身份鉴权的（您也可以替换成您期望的身份鉴权方案），所以您同样需要参考文档 [生成UserSig](#) 并给 Client 派发登录签名。

RoomService 后台源码中，`logic\im_mgr.js` 内部的 `getSig` 函数是 node.js 版本的 UserSig 生成示例代码，java 和 php 版本的我们稍后放出。

实现原理

1. 两种“通道”

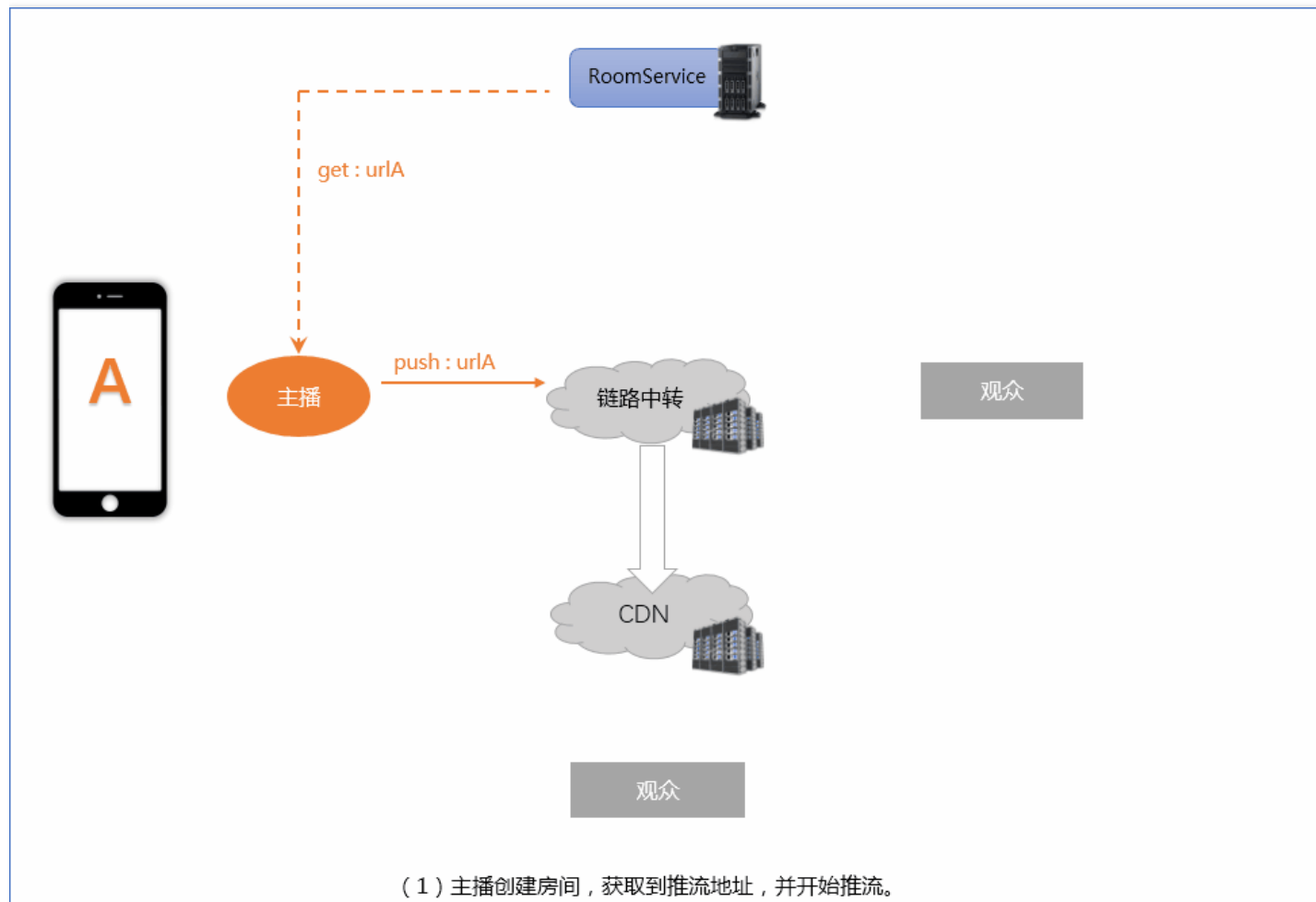
腾讯云采用了两套通道实现直播+连麦功能，其中直播采用标准的（RTMP + FLV）协议，走标准CDN线路，没有并发观看人数的限制，且带宽成本很低，但延迟一般在2s以上。连麦则采用私有的 UDP 协议，走特殊专线线路，延迟一般在500ms左右，但最多支持10人同时视频通话，且单路费用高于普通直播。



通道	直播通道	连麦通道
通讯延迟	$\geq 2s$	500ms左右
底层协议	RTMP/HTTP-FLV	私有UDP协议
价格/费用	按带宽计费	按时长计费
最高并发	无上限	≤ 10 人
TXLivePusher	setVideoQuality 为 SD、HD、FHD	setVideoQuality 为 MAIN_PUBLISHER、SUB_PUBLISHER
TXLivePlayer	PLAY_TYPE_LIVE_FLV	PLAY_TYPE_LIVE_RTMP_ACC
播放URL	普通 FLV 地址	带防盗链签名的 RTMP 地址

2. 内部原理

您完全不需要了解 LiveRoom 的内部原理便可轻松接入，但是如果您确实感兴趣，可以通过下图了解其内部运作机制。



主播 PK (LiveRoom)

最近更新时间：2018-08-06 10:31:36

功能介绍

主播 PK 是在秀场直播场景中经常使用的一种吸引热度的方式，两个分处不同房间的主播可以相互分屏连麦（视频通话），主播与主播之间的延迟可以达到 500ms 以内，而观众在不需要切换流地址的情况下，就可以在原来的 CDN 直播流中看到主播 PK 的效果。



LiveRoom

主播 PK 是 **LiveRoom** 组件的一向基本功能（LiveRoom 还支持观众同主播连麦），它分成 Client 和 Server 两个部分：

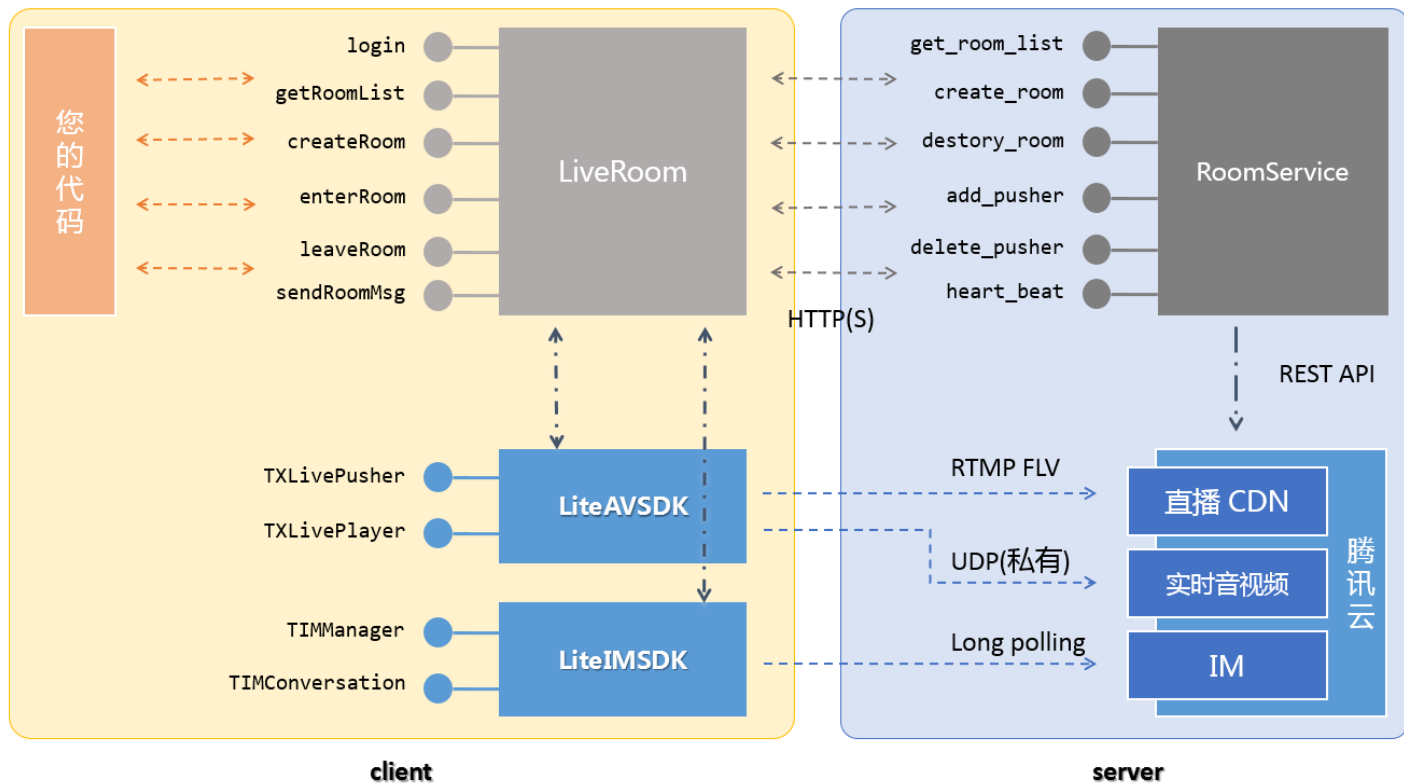
• 终端部分 (Client)

LiveRoom 组件的终端部分是对腾讯视频云 LiteAVSDK（主要用于音视频，包括 TXLivePusher、TXLivePlayer 等接口）和 LiteIMSDK（主要用于收发消息，包括 TIMManager 和 TIMConversation 等接口）的封装。直接使用 LiteAVSDK 和 LiteIMSDK 实现直播 + 主播 PK 功能是非常耗时耗力的，但通过 LiveRoom 组件，您可以直接调用 createRoom，enterRoom 和 leaveRoom 等接口就可以完成您想要的直播 + 主播 PK 功能。

• 后台部分 (Server)

RoomService 是 LiveRoom 对应的后台组件，其职责有两个：一是房间管理（直播间的增、删、改、查）和成员管理（维护房间里有几

个人正在推流，主要是连麦场景下使用，主播PK场景暂未使用）；二是对腾讯云直播服务、实时音视频服务以及 IM 云通讯服务的控制（主要通过腾讯云的后台 REST API 进行调用）。



终端对接

step1: 下载 SDK 开发包

平台	编程语言	SDK 下载	API 文档
iOS	Objective-C	DOWNLOAD	API 文档
Android	java	DOWNLOAD	API 文档

- LiteAV SDK 位于 ZIP 包解压后的 SDK 文件夹中，用于实现音视频相关功能
- LiteIM SDK 位于 ZIP 包解压后的 SDK 文件夹中，用于实现 IM 通讯相关功能
- LiveRoom 位于 ZIP 包解压后的 Demo \ liveroom 文件夹中，代码开源，方便您进行调试和定制。

step2: 登录 (login)

LiveRoom 单靠一个终端的组件无法独自运行，它依赖一个后台服务为其实现房间管理和状态协调，这个后台服务我们称之为**房间服务**（RoomService）。而要使用这个房间服务，LiveRoom 就需要先进行**登录**（login）。

阅读 [RoomService](#) 可以了解 login 函数的几个参数应该如何填写。

step3: 获取房间列表 (getRoomList)

不管是主播还是观众，都需要有一个房间列表，调用 LiveRoom 的 **getRoomList** 接口可以获取到该列表。列表中每一个房间都有其对应的 roomInfo，是在 createRoom 时传入的，推荐您将 roomInfo 定义为 json 格式，这样可以有很强的扩展性。

如果您希望使用自己的房间列表，这一步可以省略，但是需要您在 step 4 中自行指定 roomId，且全局不能有重复。

step4: 主播开播 (createRoom)

主播要开播，需要先调用 LiveRoom 的 **startLocalPreview** 接口开启本地摄像头预览，该函数需要传入一个 view 对象，该对象用于显示摄像头的视频影像。这期间 LiveRoom 会申请摄像头使用权限，同时，主播也可以对着摄像头调整一下美颜和美白的具体效果。

之后，通过调用 **createRoom** 接口，LiveRoom 会在后台的房间列表中新建一个直播间，同时主播端会进入推流模式。

参数roomId

如果您在调用 createRoom 时不填写 roomId，后台会为您分配一个 roomId，并通过 createRoom 的回调接口返回给您。如果您希望自己管理房间列表，roomId 可以由您的服务器分配，那么只需要在调用 createRoom 时填写您后台分配的 roomId 即可。

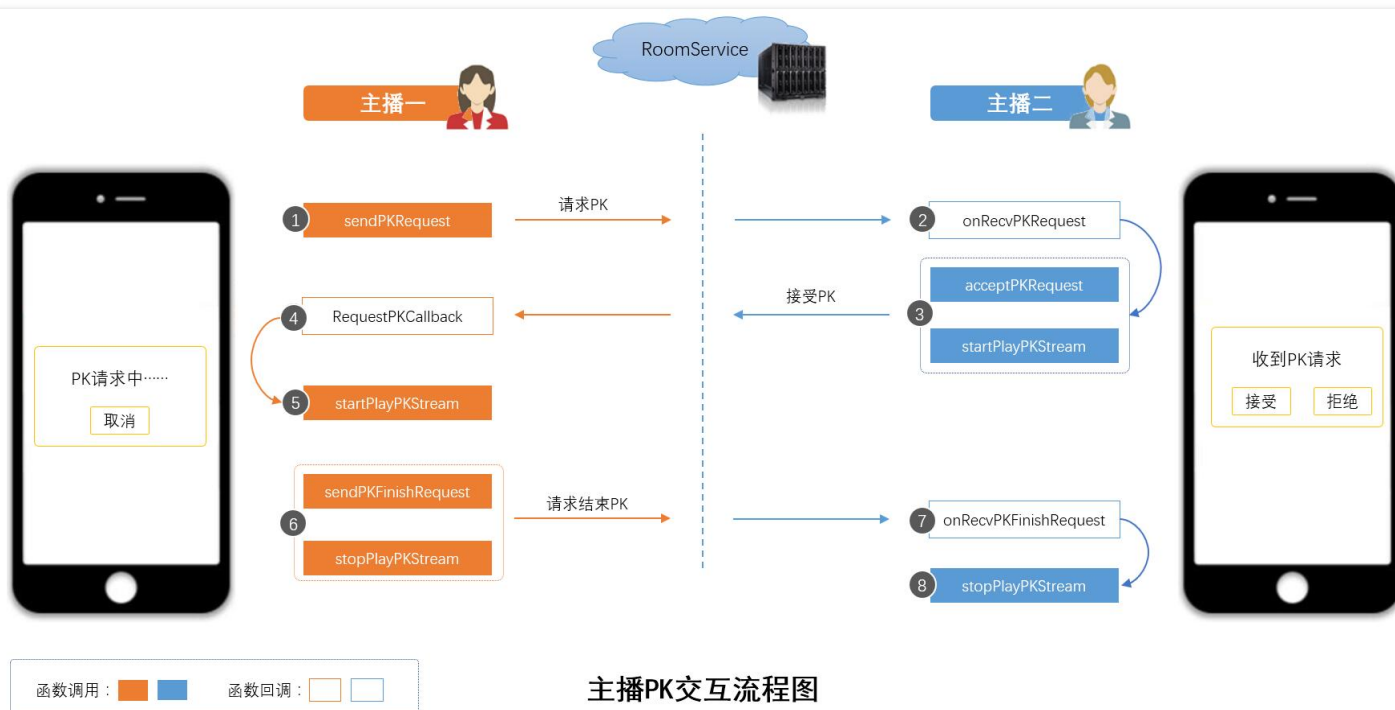
step5: 观看直播 (enterRoom)

观众通过 LiveRoom 的 **enterRoom** 接口可以进入直播间观看视频直播，enterRoom 函数需要传入一个 view 对象，用于显示直播流的视频影像。

另外，进入房间后，调用 LiveRoom 的 **getAudienceList** 接口可以获取观众列表，这里的列表不是全量数据，如果少于30 个人就全部返回，如果多于 30 个人，就只返回新进入的 30 个人。（出于性能方面的考虑，而且 UI 界面上最多能也就能放下 10 个头像。）

step6: 主播PK (sendPKRequest)

主播PK是两个房间的主播在直播的同时，互相拉取对方的视频流，建立实时视频通话互动，每个房间的观众都可以看到两个主播互动的过程，流程图如下：



a. 选择目标主播

调用 `getOnlinePusherList` 获取当前正在直播的主播列表，回调函数 `GetOnlinePusherListCallback` 返回在线主播的详细信息，包含昵称、头像、用户ID等；然后显示一个 UI 列表，以便选择一个主播进行PK。

b. 启动 PK

- 第一步（主播一）：调用 `sendPKRequest`，向主播二发起 PK 请求。
- 第二步（主播二）：会收到 `onRecvPKRequest` 回调通知，之后可以展示一个 UI 提示，询问主播二要不要接受 PK。
- 第三步（主播二）：可以调用 `acceptPKRequest` 接受 PK 请求，也可以调用 `rejectPKRequest` 拒绝 PK 请求；如果接受了 PK 请求，请同时调用 `startPlayPKStream` 播放主播一的视频流。
- 第四步（主播一）：通过 `RequestPKCallback` 可以了解到 PK 请求是否被接受。
- 第五步（主播一）：如果 PK 请求被接受，请调用 `startPlayPKStream` 播放主播二的视频流。

函数 `startPlayPKStream` 除了完成拉流播放的功能；同时会触发后台混流，即把对方主播的视频流叠加到自己的流上。普通观众不需要重新拉流，可以自动观看到两个主播PK的视频画面。

c. 结束 PK

主播PK过程中，任何一方都可以主动结束PK，假设主播一主动结束PK

- 第六步（主播一）：调用 `sendPKFinishRequest`，向主播二发起结束 PK 的请求；同时调用 `stopPlayPKStream` 结束播放主播二的视频流。
- 第七步（主播二）：会收到 `onRecvPKFinishRequest` 回调通知。
- 第八步（主播二）：调用 `stopPlayPKStream` 结束播放主播一的视频流。

函数 `stopPlayPKStream` 除了结束播放视频流，同时会取消后台混流。普通观众不需要重新拉流，可以自动切换到直播模式。

step7: 弹幕消息 (`sendMsg`)

LiveRoom 自带了消息发送接口，可以通过 `sendRoomTextMsg` 函数发送普通的文本消息（用来弹幕），也可以通过 `sendRoomCustomMsg` 发送自定义消息（用于点赞，送花等等）。

通过 `RoomListenerCallback` 里的 `onRecvRoomTextMsg` 和 `onRecvRoomCustomMsg` 可以收取聊天室里别人发来的文本消息和自定义消息。

ATTENTION

腾讯云 IM 每秒钟最多可以收取 40 条以上的消息，如果您要把所有这些消息都按照接收频率刷新到屏幕 UI 上，那您的直播体验一定是非常卡顿的，这里一定要注意刷新频率控制。

有太多的客户在测试期间顺顺利利，APP一上线就卡的不行，都是这个原因导致的。

后台对接

LiveRoom 为什么需要 login ?

LiveRoom 单靠一个终端的组件无法独自运行，它依赖一个后台服务为其实现房间管理和状态协调，这个后台服务我们称之为**房间服务**（RoomService）。而要使用这个房间服务，LiveRoom 就需要先进行**登录**（login）。

login 有很多参数需要填写，我应当如何填写这些参数呢？

如下表格中列举了三种填写方案，每种方案都有其适用场景：方案一只能用于调试；方案二适合快速上线；方案三适合自行定制；

参数名	方案一（纯测试方案）	方案二（腾讯云RoomService）	方案三（自建RoomService）
serverDomain	使用腾讯云 RoomService https://room.qcloud.com/weapp /live_room	使用腾讯云 RoomService https://room.qcloud.com/weapp /live_room，需要提前 配置	自行部署 RoomService https://[yourcompany]/weapp /live_room
sdkAppID	通过测试地址获取 https://room.qcloud.com/weapp /utils/get_login_info_debug	自行填写， 如何获取？	自行填写， 如何获取？
accType	通过测试地址获取 https://room.qcloud.com/weapp /utils/get_login_info_debug	自行填写， 如何获取？	自行填写， 如何获取？
userID	通过测试地址获取 https://room.qcloud.com/weapp /utils/get_login_info_debug	您来指定，比如 9527	您来指定，比如 9527
userSig	通过测试地址获取 https://room.qcloud.com/weapp /utils/get_login_info_debug	您的服务器生成， 如何生成？	您的服务器生成， 如何生成？
账号归属	账号为腾讯云测试账号	账号为您名下的账号	账号为您名下的账号
账号限制	每天 10：00 - 22：00 可用	无限制，但不支持定制	无限制，且您可以进行定制
适用场景	前期终端研发调试使用	产品上线初期阶段	产品进入上升期

方案一：纯调试方案

该方案使用腾讯云为方便客户调试而统一提供的测试账号，仅适合调试期间使用，每天 10：00 - 22：00 这个时间段可用。

step1. 配置RoomService

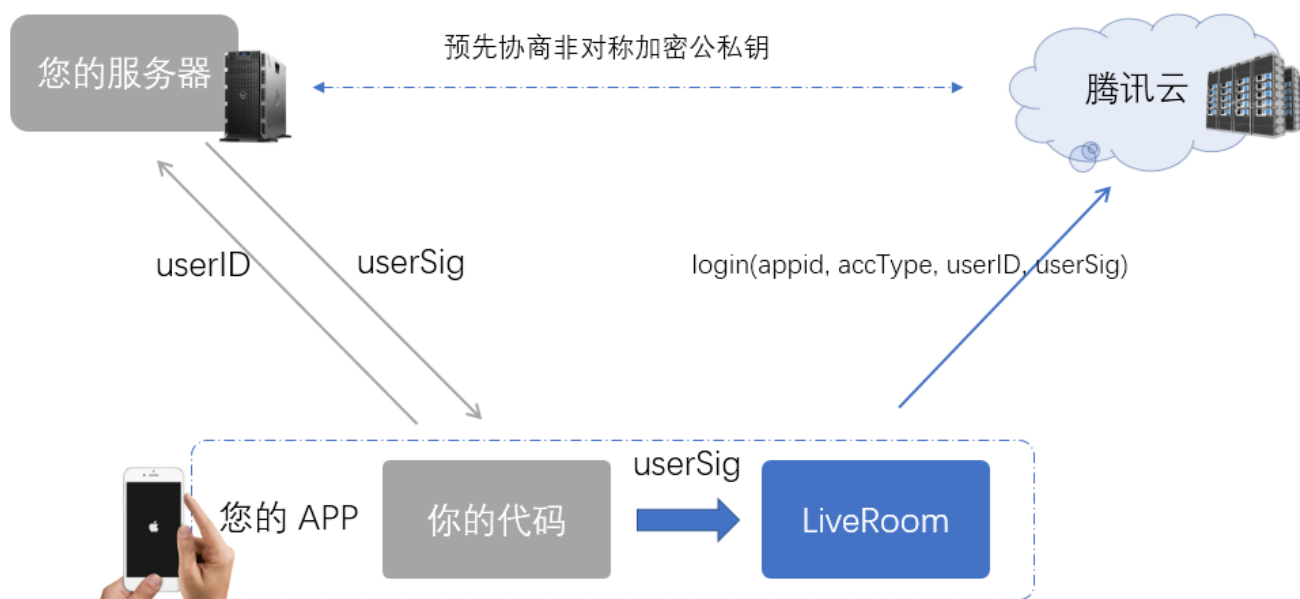
serverDomain 填写 https://room.qcloud.com/weapp/live_room 即可。

step2. 获得 login 所需参数

通过测试地址（ https://room.qcloud.com/weapp/utils/get_login_info_debug ）获取相关参数

方案二：使用腾讯云RoomService

该方案使用您自己的腾讯云账号 + 腾讯云自行部署的 RoomService 服务，所以在使用前，您需要先对 RoomService 进行配置。



step1. 配置RoomService

腾讯云 RoomService 地址为

https://room.qcloud.com/weapp/live_room

点击 [RoomTool.zip](#) 下载腾讯云 RoomService 后台配置工具，这是一个基于 Node.js 的配置工具，需要您在使用前 [安装](#) Node.js。配置工具压缩包中包含的 pdf 和 PPT 有详细的配置说明，这里仅简要概括一下各个配置项的含义和作用。

配置项	作用	获取方案
直播（live）appid	腾讯云直播服务基于 appID 区分客户身份	DOC
直播（live）APIKey	腾讯云直播服务的后台 REST API，采用 APIKey 进行安全保护	DOC
云通讯（IM）sdkAppID	腾讯云通讯服务用 sdkAppID 区分 IM 客户身份	DOC
云通讯（IM）accountType	曾用于区分 APP 类型，现仅出于兼容性原因而保留	DOC
云通讯（IM）administrator	RoomService 使用了 IM REST API 发送房间里的系统消息，而 IM REST API 接口需要您填写管理员名称。	DOC
云通讯（IM）privateKey	RoomService 使用 privateKey 用于签发管理员（administrator）的 usersig，进而能够调用 IM REST API 发送房间里的系统消息。	DOC
云通讯（IM）publicKey	RoomService 使用 publicKey 用于确认终端用户的登录身份。	DOC

step2. 获得 login 所需参数

RoomService 需要终端调用 `login(serverDomain, sdkAppID, accType, userID, userSig)` 登录成功后才能使用，其中前四个信息都可以在客户端写死，但是 UserSig 必须由您的后台服务器签发，因为让 Client 计算 UserSig 就需要将签名私钥写在终端的代码里，这会有私钥被

hack 窃取的安全风险。

RoomService 所使用的 UserSig 的签发同 IM 云通讯服务是一样的，所以同一个 UserSig 既可以用来登录 IM，又可以用来登录 RoomService，您可以参考文档 [派发UserSig](#) 进行接入。

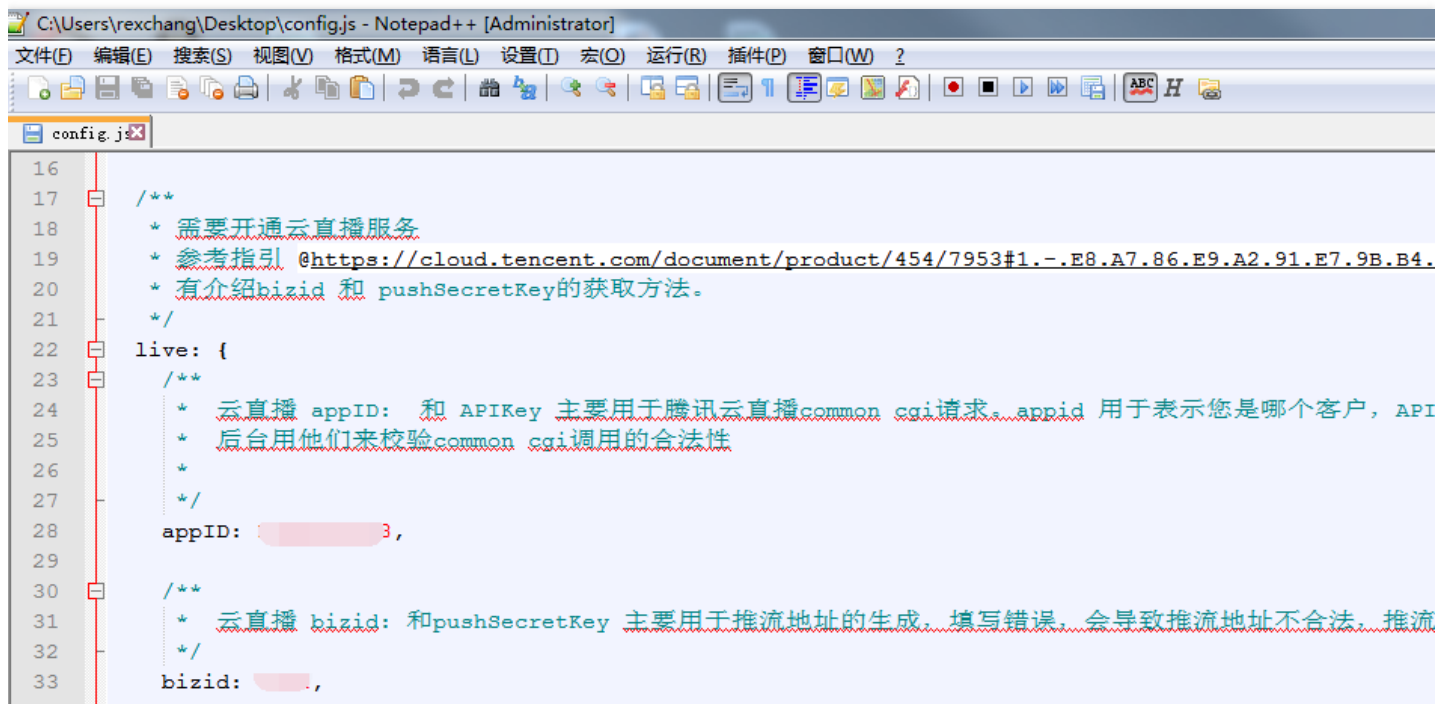
方案三：自建RoomService后台

该方案使用您自己的腾讯云账号 + 您自己部署的房间服务，所以您可以对内部的逻辑进行修改和定制。

step1. 下载源码 & 修改配置 & 部署

在 [CODE](#) 下载 RoomService 后台源码，源码包分成三个目录，其中 live_room 下的源码是您需要关注的。

下载到源码后，解压并找到 live_room 文件夹下面的 config.js 文件，这里有几个配置项需要修改。配置项跟方案一中的基本类似，可以参考方案一中的 step1 进行配置。区别在于，在方案二中您不需要用 RoomService 配置工具，而是直接修改本地源码即可。



```
16
17 /**
18  * 需要开通云直播服务
19  * 参考指引 @https://cloud.tencent.com/document/product/454/7953#1.-.E8.A7.86.E9.A2.91.E7.9B.B4.
20  * 有介绍bizid 和 pushSecretKey的获取方法。
21  */
22 live: {
23   /**
24    * 云直播 appID: 和 APIKey 主要用于腾讯云直播common cgi请求。appid 用于表示您是哪个客户，API
25    * 后台用他们来校验common cgi调用的合法性
26    *
27    */
28    appID: '3',
29
30    /**
31     * 云直播 bizid: 和pushSecretKey 主要用于推流地址的生成。填写错误，会导致推流地址不合法，推流
32     */
33    bizid: ' ',
```

配置项	作用	获取方案
直播 (live) pushSecretKey	用于计算推流 URL 的防盗链签名，必须要配置	DOC
直播 (live) APIKey	腾讯云直播服务的后台 REST API，采用 APIKey 进行安全保护	DOC

之后，您可以将其部署到您自己的后台服务器上，并将服务器的外网 URL 告知您的终端研发工程师，因为他/她在调用终端的 [login](#) 函数时需要指定 RoomService 的后台地址，例如：

```
https://[www.yourcompany.com]/weapp/live_room
```

step2. 获得 login 所需参数

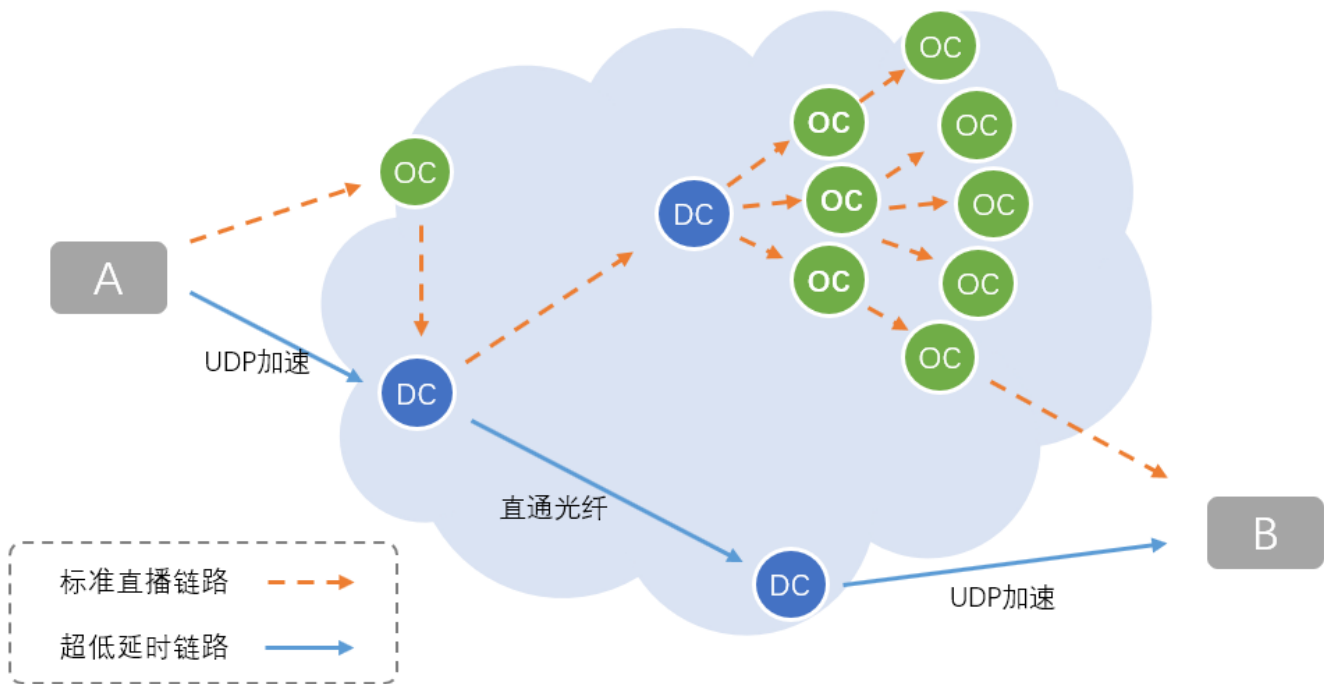
第一步完成之后，相当于您已经拥有了一个自己的 RoomService，但这个自己的 RoomService 还是基于 sdkAppID, accType, userID, userSig 这四个信息进行身份鉴权的（您也可以替换成您期望的身份鉴权方案），所以您同样需要参考文档 [生成UserSig](#) 并给 Client 派发登录签名。

RoomService 后台源码中，`logic\im_mgr.js` 内部的 `getSig` 函数是 node.js 版本的 UserSig 生成示例代码，java 和 php 版本的我们稍后放出。

实现原理

1. 两种“通道”

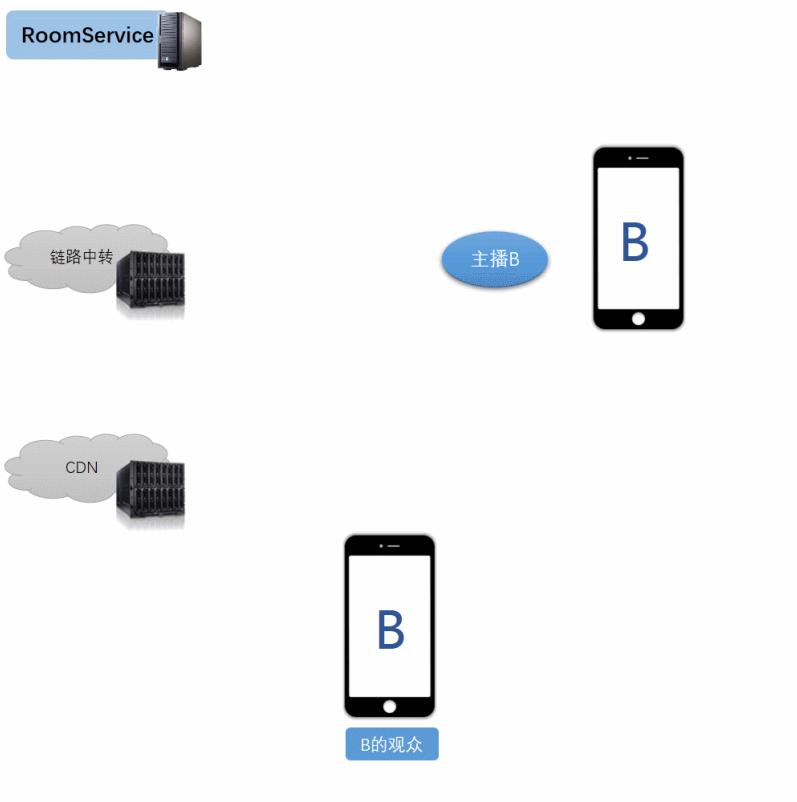
腾讯云采用了两套通道实现直播+主播PK功能，其中直播采用标准的（RTMP + FLV）协议，走标准CDN线路，没有并发观看人数的限制，且带宽成本很低，但延迟一般在2s以上。主播PK采用私有的 UDP 协议，走特殊专线线路，延迟一般在500ms左右，但最多支持10人同时视频通话，且单路费用高于普通直播。



通道	直播通道	主播PK通道
通讯延迟	$\geq 2s$	500ms左右
底层协议	RTMP/HTTP-FLV	私有UDP协议
价格/费用	按带宽计费	按时长计费
最高并发	无上限	≤ 10 人
TXLivePusher	setVideoQuality 为 SD、HD、FHD	setVideoQuality 为 MAIN_PUBLISHER
TXLivePlayer	PLAY_TYPE_LIVE_FLV	PLAY_TYPE_LIVE_RTMP_ACC
播放URL	普通 FLV 地址	带防盗链签名的 RTMP 地址

2. 内部原理

您完全不需要了解 LiveRoom 的内部原理便可轻松接入，但是如果您确实感兴趣，可以通过下图了解其内部运作机制。



视频通话 (RTCRoom)

最近更新时间：2018-07-10 14:50:54

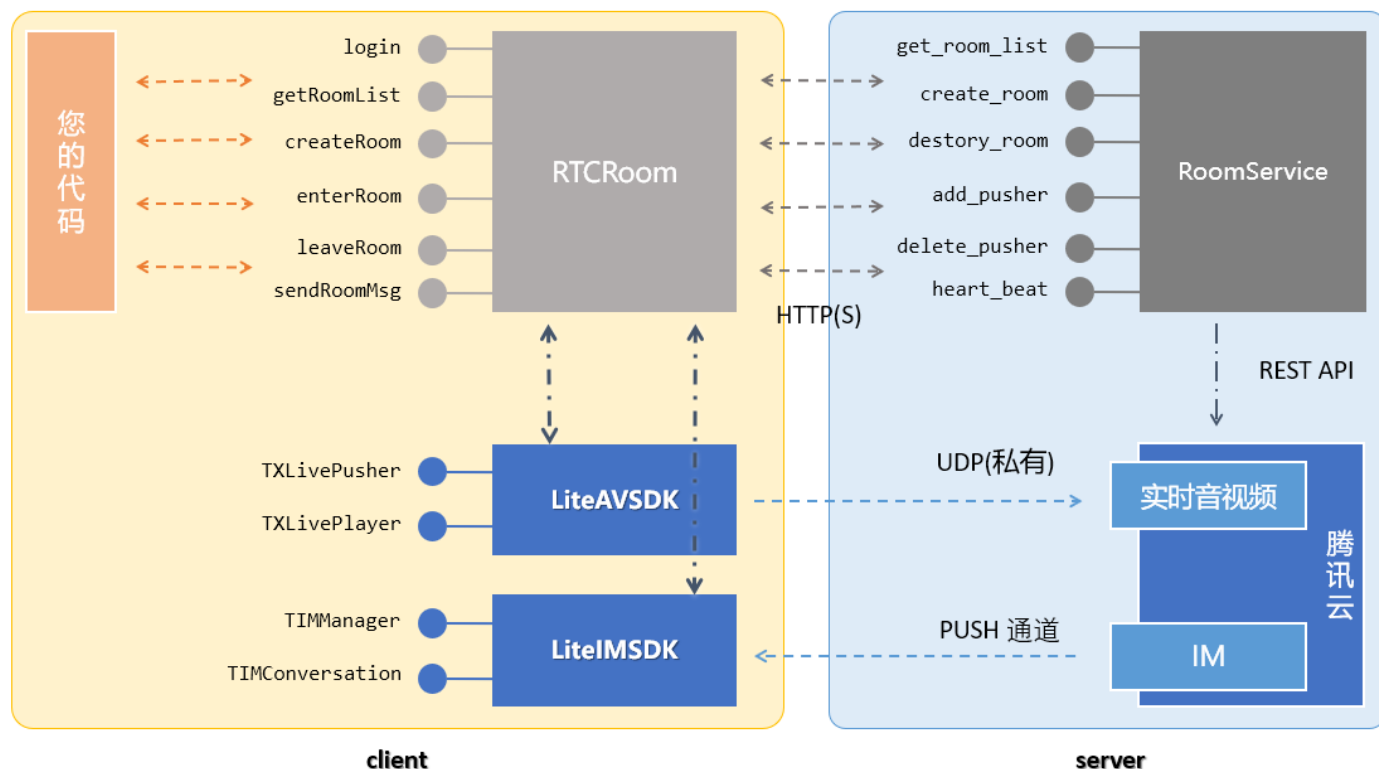
功能介绍

RTCRoom 是一个比较纯粹的视频通话解决方案，用于实现双人&多人的实时视频通话，适用场景有：在线客服、远程定损、远程开户、在线庭审 等应用场景。该方案目前有 iOS、Android、小程序、Windows 等多个平台的实现版本，切支持各平台之间的互通。



RTCRoom

RTCRoom 分成 Client 和 Server 两个部分：



• 终端部分 (Client)

RTCRoom 组件的终端部分是开源的，它是对腾讯视频云 LiteAVSDK（主要用于音视频，包括 TXLivePusher、TXLivePlayer 等接口）和 LiteIMSDK（主要用于收发消息，包括 TIMManager 和 TIMConversation 等接口）两个组件的封装。

您不需要接触 LiteAVSDK 和 LiteIMSDK 这两个较底层的 SDK，可以直接使用 RTCRoom 的 createRoom，enterRoom 和 leaveRoom 等接口就可以完成视频通话功能的搭建。

同时，它也依赖一个用于房间管理和状态协调的后台服务，这是下面要介绍的：

• 后台部分 (Server)

RoomService 是 RTCRoom 对应的后台组件（同样也是开源的），其主要代码逻辑分成两个部分：一是房间管理（直播间的增、删、改、查）和成员管理（尤其是要维护好房间里正在视频通话的成员）；二是对腾讯云实时音视频服务以及 IM 云通讯服务的控制（主要通过腾讯云的后台 REST API 进行调用）。

您可以将我们提供的源码部署到您的业务后台上，也可以使用腾讯云免费提供的 RoomService 服务。

终端对接

step1: 下载 SDK 开发包

平台	编程语言	SDK 下载	API 文档
iOS	Objective-C	DOWNLOAD	API 文档
Android	java	DOWNLOAD	API 文档
微信小程序	javascript	DOWNLOAD	API 文档

平台	编程语言	SDK 下载	API 文档
IE浏览器	javascript	DOWNLOAD	API 文档
Windows (C++)	C++	DOWNLOAD	API 文档
Windows (C#)	C#	DOWNLOAD	API 文档

- LiteAV SDK 位于 ZIP 包解压后的 SDK 文件夹中，用于实现音视频相关功能
- LiteIM SDK 位于 ZIP 包解压后的 SDK 文件夹中，用于实现 IM 通讯相关功能
- RTCRoom 位于 ZIP 包解压后的 Demo \ rtcroom 文件夹中，代码开源，方便您进行调试和定制。

step2: 登录 (login)

RTCRoom 单靠一个终端的组件无法独自运行，它依赖一个后台服务为其实现房间管理和状态协调，这个后台服务我们称之为**房间服务**（RoomService）。而使用这个房间服务，RTCRoom 就需要先进行**登录**（login）。

阅读 [RoomService](#) 可以了解 login 函数的几个参数应该如何填写。

step3: 获取房间列表

调用 RTCRoom 的 **getRoomList** 接口可以获取房间列表，列表中每一个房间都有其对应的 roomInfo，是在 createRoom 时传入的，推荐您将 roomInfo 定义为 json 格式，这样可以有很强的扩展性。

如果您希望使用自己的房间列表，这一步可以省略，但是需要您在 step 4 中自行指定 roomId，且全局不能有重复。

step4: 创建新房间

调用 RTCRoom 的 **startLocalPreview** 接口开启本地摄像头预览，该函数需要传入一个 view 对象，该对象用于显示摄像头的视频影像。这期间 RTCRoom 会申请摄像头使用权限，同时，主播也可以对着摄像头调整一下美颜和美白的具体效果。

之后，通过调用 **createRoom** 接口，RTCRoom 会在后台的房间列表中新建一个直播间，同时房间的创建者会进入推流模式。

参数 RoomID

如果您在调用 createRoom 时不填写 roomId，后台会为您分配一个 roomId，并通过 createRoom 的回调接口返回给您。如果您希望自己管理房间列表，roomId 可以由您的服务器分配，那么只需要在调用 createRoom 时填写您后台分配的 roomId 即可。

step5: 加入已有房间

通过 RTCRoom 的 **enterRoom** 接口可以进入参数 roomId 所指定的房间。如果房间里面有人，那么 RoomListenerCallback 的 **onGetPusherList** 会返回房间里已有成员的列表（pusherInfoList）。

之后，调用 **addRemoteView** 函数，并传入一个 pusherInfo 和一个 view 对象，就可以在指定的 view 上显示 pusherInfo 所指定的远程影像。

在房间里的这段时间，如果有成员加入或者离开，RoomListenerCallback 会通过 **onPusherJoin** 和 **onPusherQuit** 通知给您的代码。

step6: 收发文字消息

LiveRoom 自带了消息发送接口，可以通过 **sendRoomTextMsg** 函数发送普通的文本消息，也可以通过 **sendRoomCustomMsg** 发送自定义消息（比如传输图片的 url 等等）。

通过 RoomListenerCallback 里的 `onRecvRoomTextMsg` 和 `onRecvRoomCustomMsg` 可以收取聊天室里别人发来的文本消息和自定义消息。

后台对接

RTCRoom 为什么需要 login ?

RTCRoom 单靠一个终端的组件无法独自运行，它依赖一个后台服务为其实现房间管理和状态协调，这个后台服务我们称之为**房间服务**（RoomService）。而要使用这个房间服务，RTCRoom 就需要先进行**登录**（login）。

login 有很多参数需要填写，我应当如何填写这些参数呢？

如下表格中列举了三种填写方案，每种方案都有其适用场景：方案一适合调试；方案二适合快速上线；方案三适合自行定制；

参数名	方案一（纯测试方案）	方案二（腾讯云RoomService）	方案三（自建RoomService）
serverDomain	使用腾讯云 RoomService <code>https://room.qcloud.com/weapp/rtc_room</code>	使用腾讯云 RoomService <code>https://room.qcloud.com/weapp/rtc_room</code> ，需要提前 配置	自行部署 RoomService <code>https://[yourcompany]/weapp/rtc_room</code>
sdkAppID	通过测试地址获取 <code>https://room.qcloud.com/weapp/utils/get_login_info_debug</code>	自行填写， 如何获取？	自行填写， 如何获取？
accType	通过测试地址获取 <code>https://room.qcloud.com/weapp/utils/get_login_info_debug</code>	自行填写， 如何获取？	自行填写， 如何获取？
userID	通过测试地址获取 <code>https://room.qcloud.com/weapp/utils/get_login_info_debug</code>	您来指定，比如 9527	您来指定，比如 9527
userSig	通过测试地址获取 <code>https://room.qcloud.com/weapp/utils/get_login_info_debug</code>	您的服务器生成， 如何生成？	您的服务器生成， 如何生成？
账号归属	账号为腾讯云测试账号	账号为您名下的账号	账号为您名下的账号
账号限制	每天 10 : 00 - 22 : 00 可用	无限制，但不支持定制	无限制，且您可以进行定制
适用场景	前期终端研发调试使用	产品上线初期阶段	产品进入上升期

方案一：纯调试方案

该方案使用腾讯云为方便客户调试而统一提供的测试账号，仅适合调试期间使用，每天 10 : 00 - 22 : 00 这个时间段可用。

step1. 配置RoomService

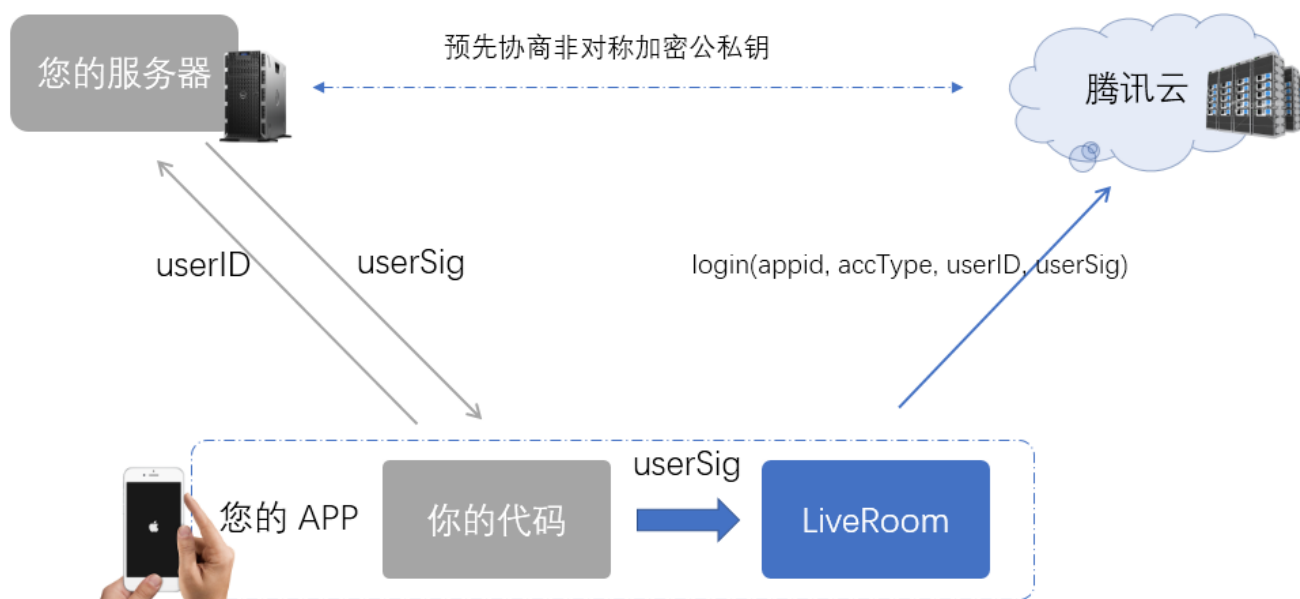
serverDomain 填写 `https://room.qcloud.com/weapp/rtc_room` 即可。

step2. 获得 login 所需参数

通过测试地址（`https://room.qcloud.com/weapp/utils/get_login_info_debug`）获取相关参数

方案二：使用腾讯云RoomService

该方案使用您自己的腾讯云账号 + 腾讯云自行部署的 RoomService 服务，所以在使用前，您需要先对 RoomService 进行配置。



step1. 配置RoomService

腾讯云 RoomService 地址为

https://room.qcloud.com/weapp/rtc_room

点击 [RoomTool.zip](#) 下载腾讯云 RoomService 后台配置工具，这是一个基于 Node.js 的配置工具，需要您在使用前 [安装](#) Node.js。配置工具压缩包中包含的 pdf 和 PPT 有详细的配置说明，这里仅简要概括一下各个配置项的含义和作用。

配置项	作用	获取方案
直播（live）appid	腾讯云直播服务基于 appID 区分客户身份	DOC
直播（live）APIKey	腾讯云直播服务的后台 REST API，采用 APIKey 进行安全保护	DOC
云通讯（IM）sdkAppID	腾讯云通讯服务用 sdkAppID 区分 IM 客户身份	DOC
云通讯（IM）accountType	曾用于区分 APP 类型，现仅出于兼容性原因而保留	DOC
云通讯（IM）administrator	RoomService 使用了 IM REST API 发送房间里的系统消息，而 IM REST API 接口需要您填写管理员名称。	DOC
云通讯（IM）privateKey	RoomService 使用 privateKey 用于签发管理员（administrator）的 usersig，进而能够调用 IM REST API 发送房间里的系统消息。	DOC
云通讯（IM）publicKey	RoomService 使用 publicKey 用于确认终端用户的登录身份。	DOC

step2. 获得 login 所需参数

RoomService 需要终端调用 `login(serverDomain, sdkAppID, accType, userID, userSig)` 登录成功后才能使用，其中前四个信息都可以在客户端写死，但是 UserSig 必须由您的后台服务器签发，因为让 Client 计算 UserSig 就需要将签名私钥写在终端的代码里，这会有私钥被

hack 窃取的安全风险。

RoomService 所使用的 UserSig 的签发同 IM 云通讯服务是一样的，所以同一个 UserSig 既可以用来登录 IM，又可以用来登录 RoomService，您可以参考文档 [派发UserSig](#) 进行接入。

方案三：自建RoomService后台

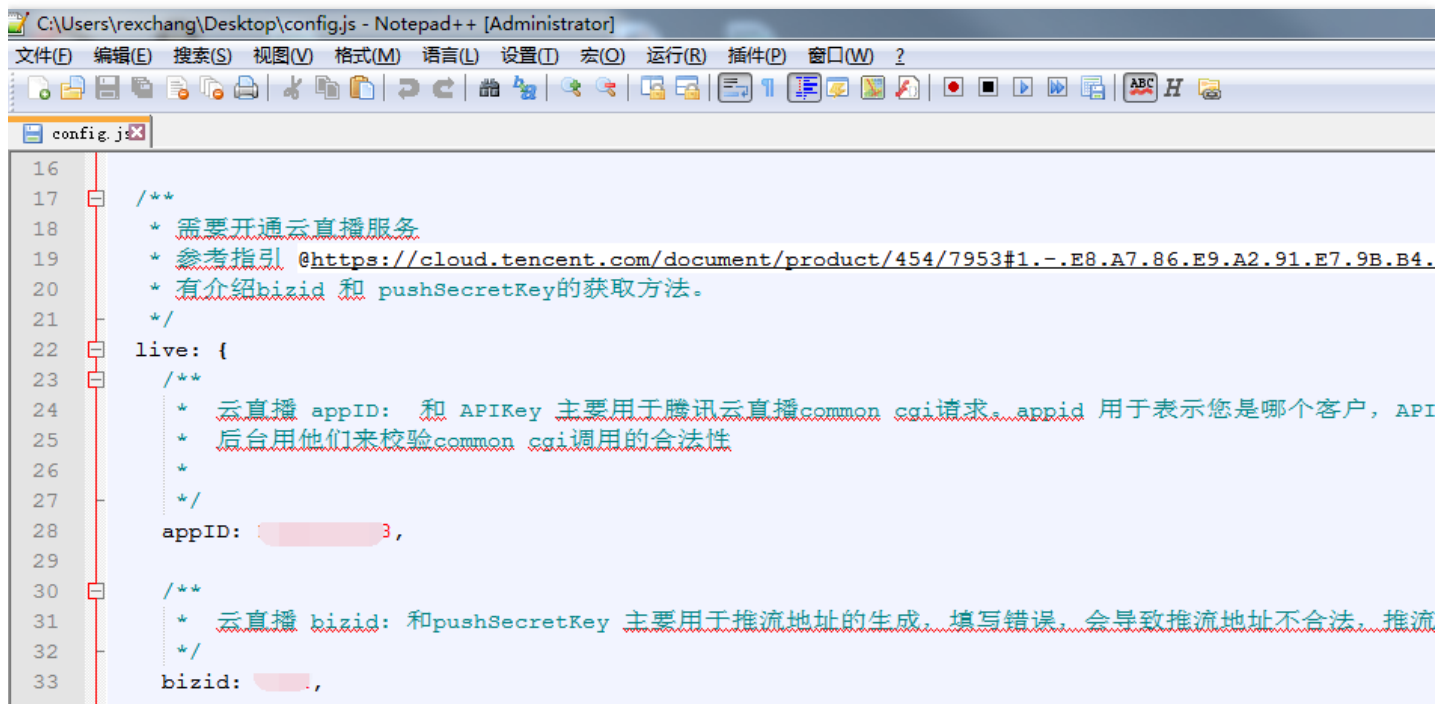
该方案使用您自己的腾讯云账号 + 您自己部署的房间服务，所以您可以对内部的逻辑进行修改和定制。

step1. 下载源码 & 修改配置 & 部署

在 [CODE](#) 下载 RoomService 后台源码，源码包分成三个目录，其中 rtc_room 下的源码是您需要关注的。

目前我们仅完成了 node.js 版本的源码，php 和 java 版本的源码还在准备中，盼您关注我们的最新更新动态 ...

下载到源码后，解压并找到 rtc_room 文件夹下面的 config.js 文件，这里有几个配置项需要修改。配置项跟方案一中的基本类似，可以参考方案一中的 step1 进行配置。区别在于，在方案二中您不需要用 RoomService 配置工具，而是直接修改本地源码即可。



```
16
17 /**
18  * 需要开通云直播服务
19  * 参考指引 @https://cloud.tencent.com/document/product/454/7953#1.-.E8.A7.86.E9.A2.91.E7.9B.B4.
20  * 有介绍bizid 和 pushSecretKey的获取方法。
21  */
22 live: {
23   /**
24    * 云直播 appID: 和 APIKey 主要用于腾讯云直播common cgi请求。appid 用于表示您是哪个客户，API
25    * 后台用他们来校验common cgi调用的合法性
26    *
27    */
28    appID: 3,
29
30    /**
31     * 云直播 bizid: 和pushSecretKey 主要用于推流地址的生成。填写错误，会导致推流地址不合法。推流
32     */
33    bizid: ,
```

配置项	作用	获取方案
直播 (live) pushSecretKey	用于计算推流 URL 的防盗链签名，必须要配置	DOC
直播 (live) APIKey	腾讯云直播服务的后台 REST API，采用 APIKey 进行安全保护	DOC

之后，您可以将其部署到您自己的后台服务器上，并将服务器的外网 URL 告知您的终端研发工程师，因为他/她在调用终端的 [login](#) 函数时需要指定 RoomService 的后台地址，例如：

```
https://[www.yourcompany.com]/weapp/rtc_room
```

step2. 获得 login 所需参数

第一步完成之后，相当于您已经拥有了一个自己的 RoomService，但这个自己的 RoomService 还是基于 `sdkAppID`, `accType`, `userID`, `userSig` 这四个信息进行身份鉴权的（您也可以替换成您期望的身份鉴权方案），所以您同样需要参考文档 [生成UserSig](#) 并给 Client 派发登录签名。

RoomService 后台源码中，`logic\im_mgr.js` 内部的 `getSig` 函数是 node.js 版本的 UserSig 生成示例代码，java 和 php 版本的我们稍后放出。

直播答题（冲顶模式）

最近更新时间：2018-08-29 18:29:40

效果体验

Demo安装

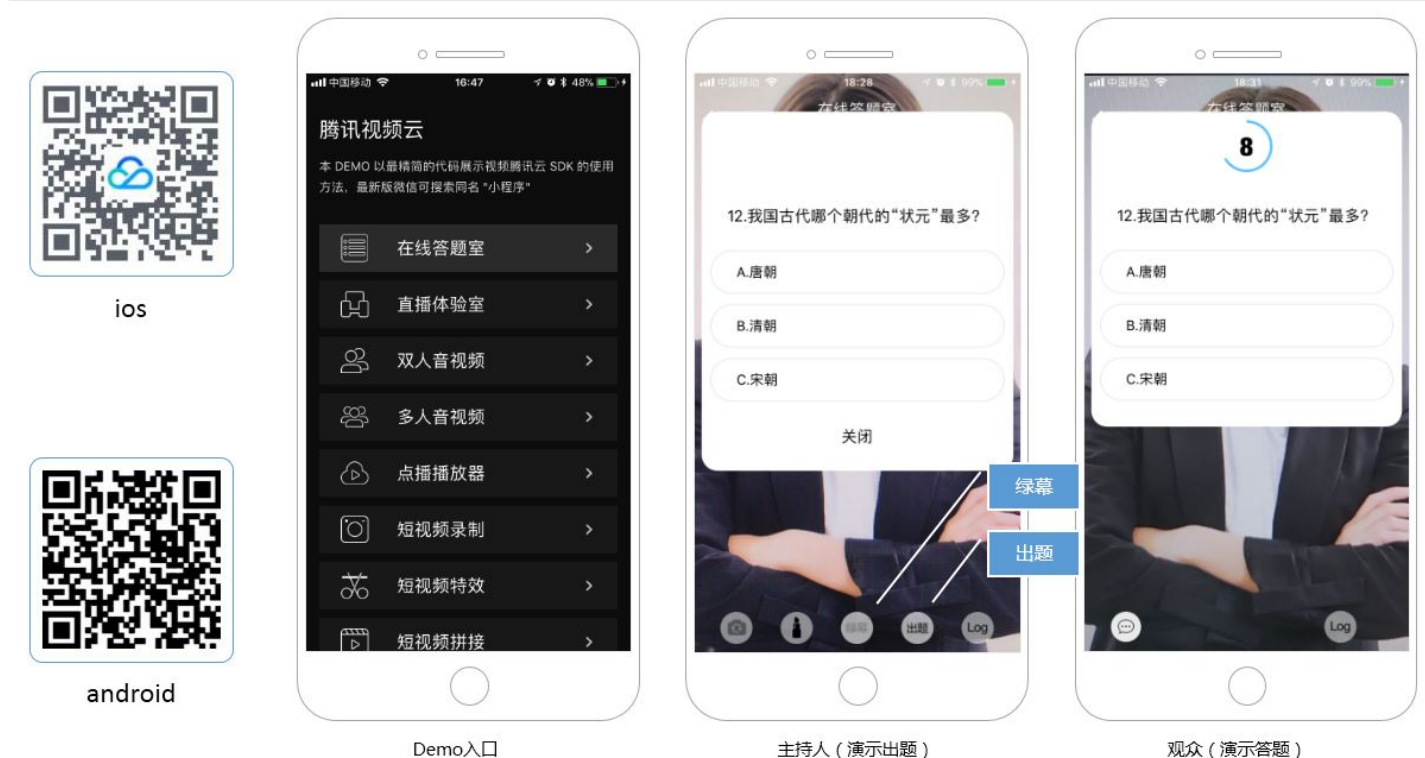
下载安装我们的 Demo 可以体验在线答题的演示效果，目前暂时只有两种可体验的方式，每一种都能做到“音-画-题”完美同步。

iOS平台 (ipa)	Android平台(apk)	Obs Studio 定制版本 (exe)
iOS	Android	Obs Studio

体验方式一：在线答题室

创建房间即可体验主持人一端的效果；进入房间即可体验观众端的效果。

- iPhone 平台的 Demo 采用了企业签名方式，请先到“设置-通用-设备管理”里，添加信任证书。
- Demo 仅作演示之用，用来展示腾讯云的技术能力，不代表最终产品形态，接入还需认真阅读此文档。



体验方式二：Obs Studio

这种体验方式是直接使用腾讯云改造的 [Obs Studio](#) 进行推流：

- 我们在“工具”菜单栏里面加了一个“题目分发”按钮，可以直接将题目打入直播流中，题目可以预先编辑在 ini 文件中。
- 使用 iOS 或者 Android 端的演示App，选择 **答题播放器** 功能，就可以体验观众端题目收取的效果了。



SDK下载

- **LiteAVSDK (4.1.3173)**

用于 RTMP 推流和 FLV 播放，其中 Smart 版包含推流和播放两个功能，LivePlay 版仅包含 FLV 播放功能。

操作系统	下载链接	RTMP推流	RTMP播放	FLV播放	备注说明
iOS	DOWNLOAD	YES	YES	YES	SDK 和 Demo 源码均在压缩包中
	DOWNLOAD	NO	YES	YES	体积增量：700K，SDK 和 Demo 源码均在压缩包中
Android	DOWNLOAD	YES	YES	YES	SDK 和 Demo 源码均在压缩包中
	DOWNLOAD	NO	YES	YES	jar增量：350K，SDK 和

Demo 源码均在压缩包中

• LiteIMSDK (1.3.0.130)

用于实现聊天室和弹幕等相关功能，此处下载的为精简版，您也可以到[腾讯云通讯](#)（IM）官网下载完整功能版。

操作系统	下载链接	备注说明
iOS	DOWNLOAD	此为精简版，体积增量为 1.74M
Android	DOWNLOAD	此为精简版，体积增量为 670k

我们的优势

• 精准的“音题画”同步

腾讯云 SDK 和云端均支持在直播流中插入 **题目** 或 **时间同步信令**，可以实现声音、画面和题目弹出的完美同步。

• 超低的观众端延迟差

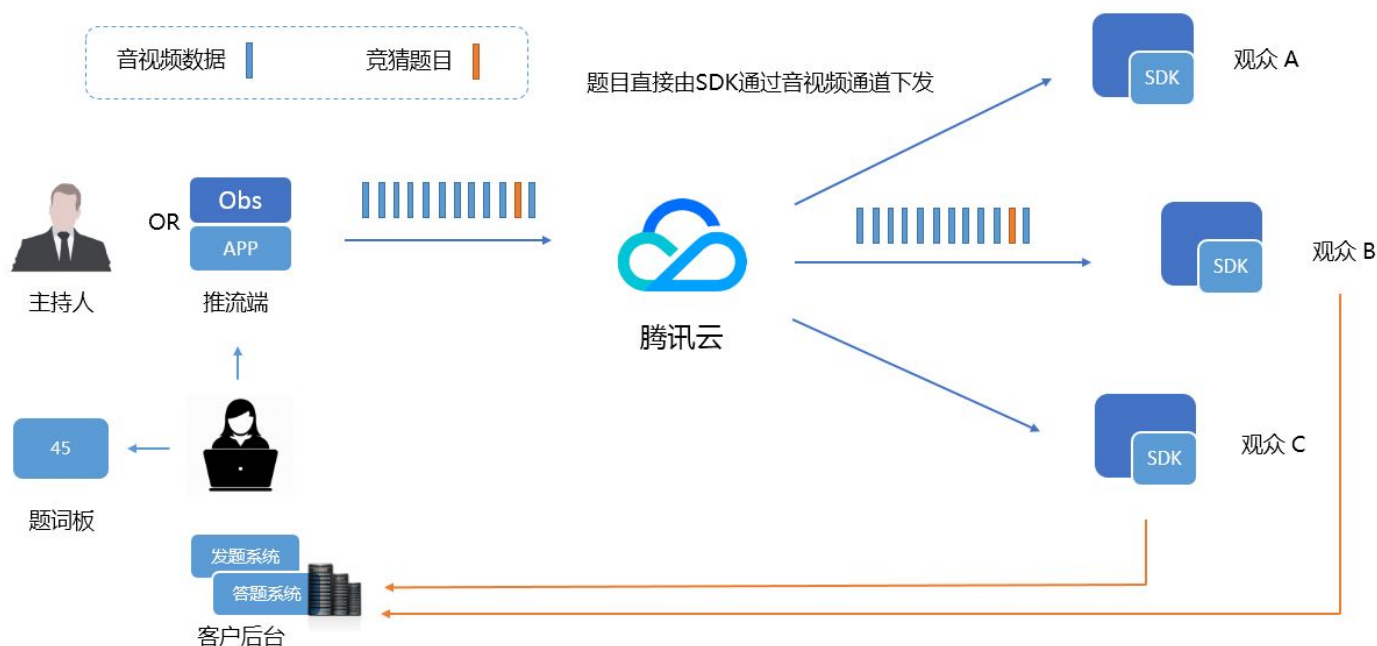
腾讯云 SDK 的 [极速播放模式](#) 所支持的延迟修正技术，可以让观众与观众之间延迟差在 1s 以内，从而让观众的答题同步性得到保证。

• 支持微信小程序接入

腾讯云 SDK 已经默认打包在微信版本中，并以 `<live-player>` 标签的形式对外提供，设置 mode 为 live 模式，并将 min-cache 和 max-cache 都设置为 1，即可实现非常低延迟的播放效果。

方案解读

方案一：题目透传方案



原理描述

• 消息发送 (Obs) :

如果你是在演播室使用 Obs 推流，可以直接使用腾讯云改造的 [Obs Studio](#) 替换现有的 Obs 软件，-我们在“工具”菜单栏里面加了一个“题目分发”按钮，可以直接将题目打入直播流中，题目可以预先编辑在 ini 文件中。

• 消息发送 (APP) :

如果想要简单使用 App 进行推流，可以使用腾讯云终端 SDK 的 TXLivePusher 的 sendMessage 方法，该方案能将一段buffer塞到 RTMP流中（buffer的最大长度限制为10K）。

```
// iOS 示例代码
[_answerPusher sendMessage:[mesg dataUsingEncoding:NSUTF8StringEncoding]];

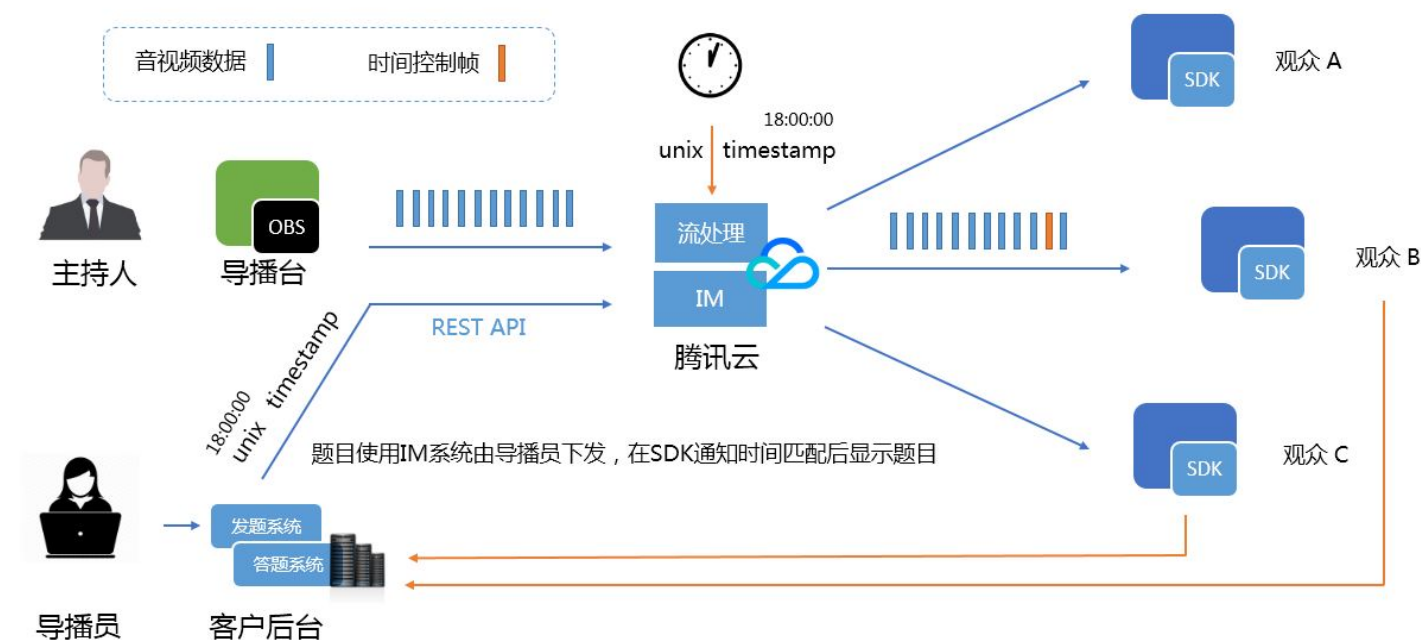
// Android 示例代码
mTXLivePusher.sendMessage(questionInfo.getBytes("UTF-8"));
```

• 消息接收 :

利用腾讯云 SDK 的 TXLivePlayer 的 onPlayEvent (PLAY_EVT_GET_MESSAGE : 2012) 功能，可以在播放器播放到指定画面的时候，同步地将 MESSAGE 通知给您的 APP，从而将题目同步地扩散到海量的观众端。

消息接收的接入方案可以参考我们的接入文档 ([iOS平台](#) | [Android平台](#))

方案二：NTP时间同步方案



原理描述

1. 腾讯云会每 1 帧在您的直播流中实时插入经过 NTP 校时的国际标准时间戳。
2. 演播室的导播员根据主持人的出题节奏，在合适的时间控制发题，发题系统会在每次下发的题目中带上当时的国际标准时间。
3. SDK 在播放这种打入时间戳的视频流的时候，会定时通知您的 APP 当前 SDK 所播放的画面是在什么时间录制下来的（因为导播台到云端一般都会有一个固定的延迟，需要您提前做一个误差校调）。
4. 您的 APP 可以根据 SDK 的时间通知（即当前画面是什么时间录制的），按需显示指定的题目即可。

总结一下：方案二跟方案一的最大区别就在于题目的扩散方式，方案二的核心思路是让题目先通过IM通道快速送达观众端APP，在观众端先缓存下来，等待播放器通知了预期的 NTP 时间戳之后，再把题目显示出来。

方案三：小程序解决方案

方案一和方案二都可以实现“音-话-题”完美同步，但是相比于这种体验上的小优化，APP的扩散能力则是更加重要的。小程序恰恰提供了APP可以病毒式传播的能力。

腾讯云 SDK 已经默认打包在微信版本中，并以 `<live-player>` 标签的形式对外提供，如果：

- 采用 flv 播放地址
- 设置 mode 为 live 模式
- 将 min-cache 和 max-cache 都设置为 1
- 推流端 gop 为 1

那么就可以实现非常理想的播放延迟，并能够让各观众之间的延迟误差在1s以内，虽然无法做到“音-话-题”分毫不差（在微信打包视频云 SDK 的时候，在线答题模式还没有兴起，所以不支持在音视频流中夹带消息），但也能达到几乎可以媲美APP的体验效果。

剩下的事情，就是通过小程序的 websocket 通道，或者我们的 webim 解决方案，向小程序发题就可以了。



接入攻略（方案一）

步骤一：开通腾讯云直播服务

联系我们开通腾讯云 [直播服务](#)，如果您时间比较急，可以 400 客服电话我们加急审核。

步骤二：获取推流URL

想要简单获取一个推流URL，可以参考文档：[快速获得地址](#)。

想要了解推流地址和直播间ID的关系，可以参考文档：[后台自动拼装](#)。

想要了解如果保护自己的推流地址不被盗用，可以参考文档：[防盗链签名](#)。

步骤三：获取播放URL

播放URL和推流URL是——映射的，映射规则参考下面的图示您就能明白了：

过期时间: 2018-01-12 23:59:59

直播码: 3891_ test

生成推流地址

推流地址: rtmp://3891.livepush.myqcloud.com/live/3891_test?bizid=3891&txSecret=1d2f4f66920c707aa01e5d327c725555&txTime=5A58DB7F

播放地址 (RTMP): rtmp://3891.liveplay.myqcloud.com/live/3891_test

播放地址 (FLV): http://3891.liveplay.myqcloud.com/live/3891_test.flv

播放地址 (HLS): http://3891.liveplay.myqcloud.com/live/3891_test.m3u8

请务必使用 **FLV** 格式的播放地址，RTMP在高并发场景下容易出现卡顿问题。

步骤四：配置推流端

如果您是使用APP进行推流，直接参考文档（[iOS](#) | [Android](#)）。

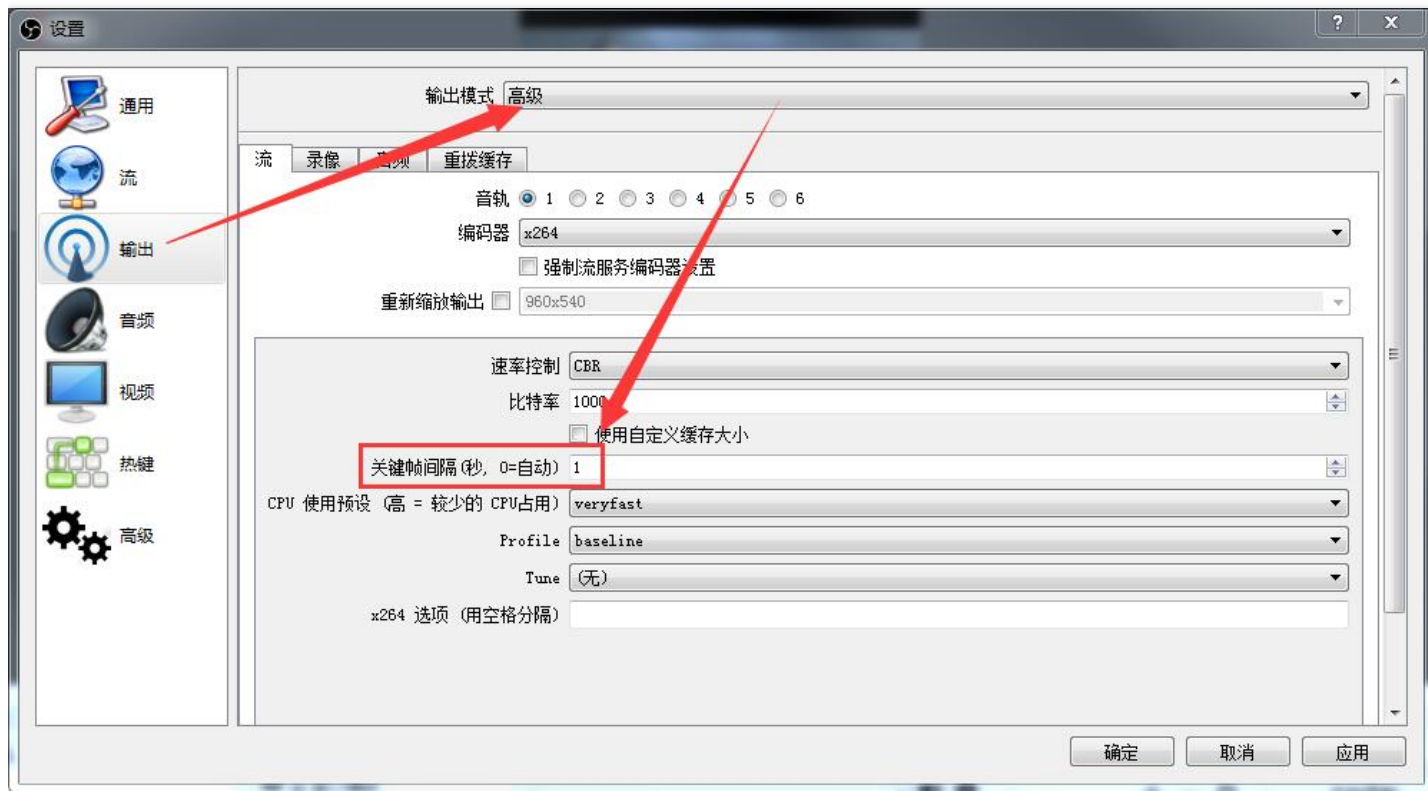
如果您是使用Obs推流，请注意以下几个重要设置项：

I帧间隔（GOP）

一般直播台的接入方式有两种：OBS Studio 推流或者编码盒推流，这两种推流工具均有比较成熟的设置接口。建议将 GOP（也叫关键帧间隔）设置为 1s，这样可以让观众端的延迟差异非常小。

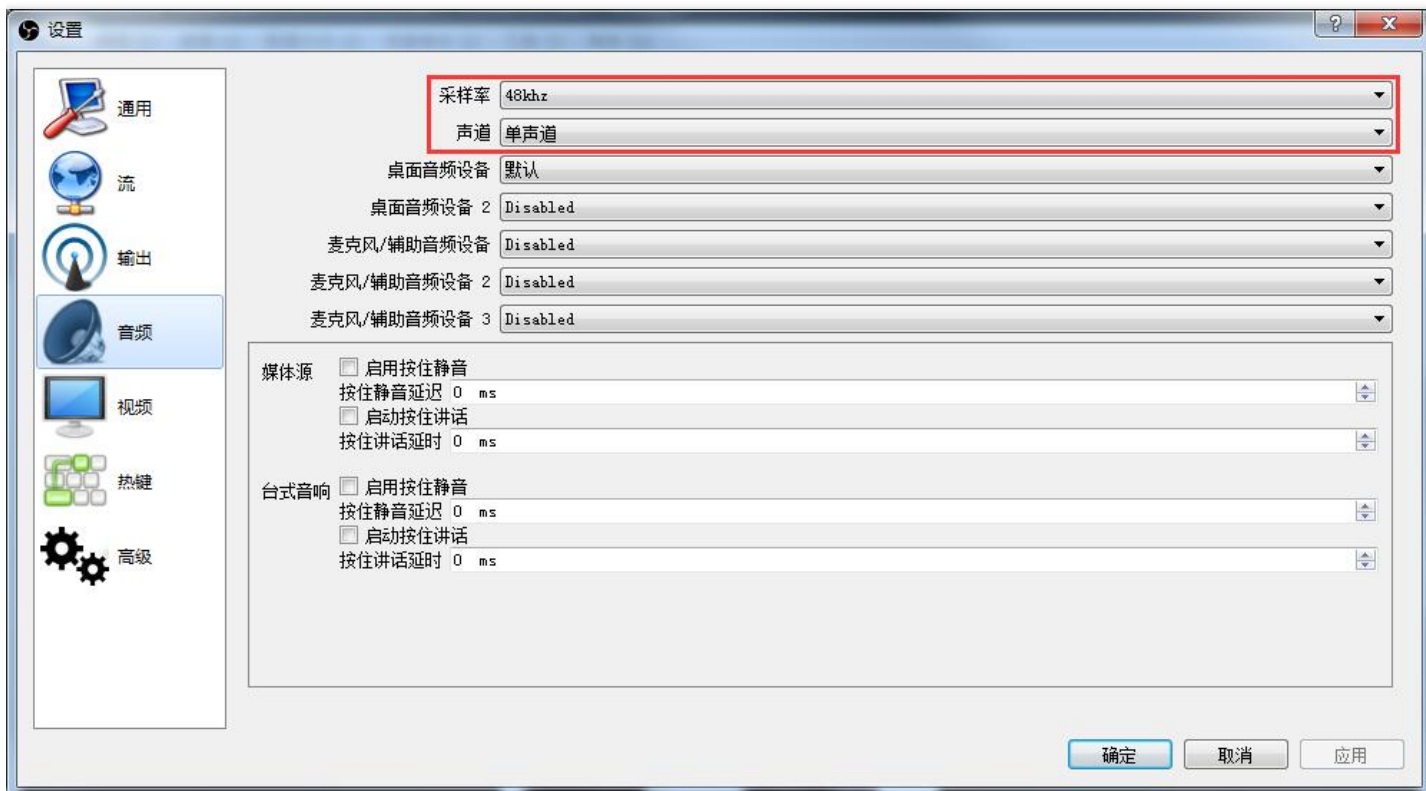
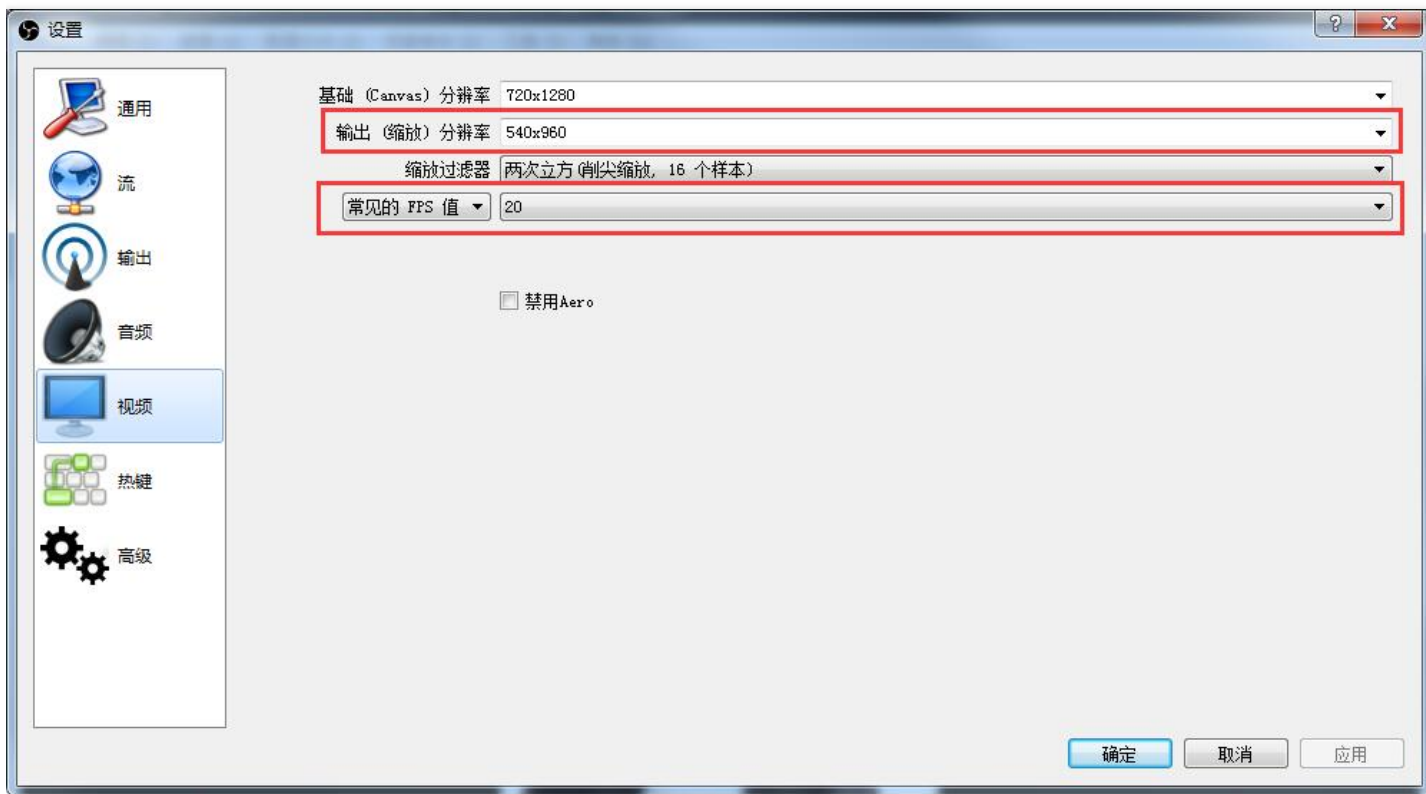
x264 的 gop 设置对编码效率的影响不是很大，但对延迟的影响非常大：gop越大，服务器缓存越多。由于 SDK 的延迟修正需要一个修正时间，如果 gop 太大，对于刚进入的观众会有很大的影响。

如下是Obs Studio 设置关键帧间隔的图示：



编码参数

推荐推荐配置	分辨率	视频比特率	帧率	声道数	采样率	音频比特率
优先画质	540x960	1000kbps	25	1	48k	72kbps
优先成本	360x640	600kbps	20	1	48k	72kbps



步骤五：对接播放器

1. 下载文档第二部分中列出的 [SDK](#) 版本。

2. 参考接入文档（[iOS](#) | [Android](#)）完成播放器的接入。两个平台全部完成，大概需要 0.5 天的工作量。

3. 修改默认配置

由于 SDK 的默认配置为普通直播场景，所以需要修改配置，操作方法如下：

```
//iOS源码
TXLivePlayConfig *config = [[TXLivePlayConfig alloc] init];
TXLivePlayer *player = [[TXLivePlayer alloc] init];
//
//开启消息接受,收不到消息的话就是没打开这个（默认:关）
config.enableMessage = YES;
//
//设置延迟平衡点为2s(考虑到云端和推流端引入的延迟，实际延迟为3s多，SDK推流：3s, obs推流：4-5秒)
config.bAutoAdjustCacheTime = YES;
config.maxAutoAdjustCacheTime = 2;
config.minAutoAdjustCacheTime = 2;
config.cacheTime = 2;
config.connectRetryCount = 3;
config.connectRetryInterval = 3;
config.enableAEC = NO;
//先setConfig再startPlay
[player setConfig:config];

//Android源码
mTXLivePlayConfig = new TXLivePlayConfig();
mTXLivePlayer = new TXLivePlayer(context);
//
//开启消息接收,收不到消息的话就是没打开这个（默认:关）
mTXLivePlayConfig.setEnableMessage(true);
//
//设置延迟平衡点为2s(考虑到云端和推流端引入的延迟，实际延迟为3s多，SDK推流：3s, obs推流：4-5秒)
mTXLivePlayConfig.setAutoAdjustCacheTime(true);
mTXLivePlayConfig.setCacheTime(2.0f);
mTXLivePlayConfig.setMaxAutoAdjustCacheTime(2.0f);
mTXLivePlayConfig.setMinAutoAdjustCacheTime(2.0f);
//
//先setConfig再startPlay
mTXLivePlayer.setConfig(mTXLivePlayConfig);
```

4. 请务必使用 **FLV** 格式的播放地址，RTMP在高并发场景下容易出现卡顿问题。

步骤六：题目的扩散

- 如果是用APP发题，只需要参考TXLivePusher 的 sendMessage 调用方法即可，参考文档见（[iOS](#) | [Android](#)）。
- 如果是用我们的定制版 [Obs Studio](#) 发题，那么可以使用本地ini文件提前把题目编辑好，由导播员在合适的时间将题目扩散出去。

可靠性评估

有些客户可能会担心：音视频通道本身不太稳定，那么万一卡了或者视频数据丢了，会不会导致观众端看不到题目。

- 首先，直播的音视频数据丢帧是确定的按照gop为单位来丢帧的，如果gop=1，那么每次会丢1s的音视频数据

- 其次，以腾讯云目前的节点部署情况来看，视频卡顿90%以上都以由于观众端网速不足引起的，这种情况下其他的网络通讯也不会很流畅。

所以，解决这个问题的办法就是每秒钟（gop设置为1s的前提条件下）发送一次题目消息，相同的题号在观众端剔除，这样就可以避免偶尔的音视频卡顿对题目到达可靠性的影响。

步骤七：接收题目消息

在我们的推流 APP Demo 以及定制版本的 Obs Studio 中，我们以 json 格式将题目组织成一段buffer，并将其夹在音视频流中送出去。

获取到这个buffer以后，您就可以将其解析出来，并完成对应的 UI 展示。如果你需要调整json格式以支持更多的定制型，请修改源码或者联系我们。

- TXLivePlayConfig 中的 **enableMessage** 开关置为 **YES**。
- TXLivePlayer 通过 **TXLivePlayListener** 监听消息，消息编号：**PLAY_EVT_GET_MESSAGE (2012)**

```
// iOS 的代码
-(void) onPlayEvent:(int)EvtID withParam:(NSDictionary *)param {
    [self asyncRun:^(
        if (EvtID == PLAY_EVT_GET_MESSAGE) {
            dispatch_async(dispatch_get_main_queue(), ^{ //抛到主线程，避免线程安全问题
                if ([_delegate respondsToSelector:@selector(onPlayerMessage:)]) {
                    [_delegate onPlayerMessage:param[@"EVT_GET_MSG"]];
                }
            });
        }
    )];
}

//Android 示例代码
mTXLivePlayer.setPlayListener(new ITXLivePlayListener() {
    @Override
    public void onPlayEvent(int event, Bundle param) {
        if (event == TXLiveConstants.PLAY_ERR_NET_DISCONNECT) {
            roomListenerCallback.onDebugLog("[AnswerRoom] 拉流失败：网络断开");
            roomListenerCallback.onError(-1, "网络断开，拉流失败");
        }
        else if (event == TXLiveConstants.PLAY_EVT_GET_MESSAGE) {
            String msg = null;
            try {
                msg = new String(param.getBytes(TXLiveConstants.EVT_GET_MSG), "UTF-8");
                roomListenerCallback.onRecvAnswerMsg(msg);
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }
        }
    }
});
```

步骤八：开发答题系统

由于腾讯云 PAAS 的定位，所以对于跟客户业务绑定比较紧密的答题和支付系统，我们就不涉及了，需要您来参与开发。

这里普遍采用的方案是：将客户的答案以 HTTP(S) 请求方式汇总到答题服务器上，只是实现过程需要注意解决瞬间高并发的请求压力。

有客户可能会问，IM系统是否适合用来答题，这里目前看是不适合的，因为IM系统的主要能力在于消息扩散，而答题的主要目标是信息的收拢。

步骤九：答题结果显示

一般题目出来一段时间以后，就会进入闭题状态。这时，答题系统会将结果进行统计汇总，并将汇总结果下发给观众。

如果是用我们的定制版 [Obs Studio](#) 扩散结果，搭一个简单的服务器，并提供一个http接口，然后按照我们约定的json格式跟Obs进行题目、答案以及人数的通讯。从而实现题目和答案的分发。

接入攻略（方案二）

步骤一：开通腾讯云直播服务

同方案一，不再赘述。

步骤二：获取推流URL & 加注NTP时间戳

参考方案一，与方案一不同之处在于，推流URL需要额外增加一个参数：

加注NTP时间戳

在推流URL之后添加参数 `&txAddTimestamp=2` 参数（之前的`txAddTimestamp=1`会在小程序上遭遇播放黑屏问题），服务器会每隔1s向您的直播流中打入一个带有国际标准时间（误差在100ms以内）的SEI时间戳。如果您使用我们的播放器播放此视频流，那么就会每隔一秒钟收到一次代表当前画面NTP时间的消息通知。

步骤三：获取播放URL

同方案一，不再赘述。

步骤四：配置推流端

同方案一，不再赘述。

步骤五：对接播放器

参考方案一，与方案一不同之处在于，此时获取的message不再是一个json，而是一个8字节的64位时间戳。

```
long timeStamp = byteArrayToInt(param.getBytes(TXLiveConstants.EVT_GET_MSG));
/**
 * 将8字节的byte数组转成一个long值
 */
public static long byteArrayToInt(byte[] byteArray) {
    byte[] a = new byte[8];
    int i = a.length - 1, j = byteArray.length - 1;
    for (; i >= 0; i--, j--) { // 从b的尾部(即int值的低位)开始copy数据
        if (j >= 0)
            a[i] = byteArray[j];
        else
            a[i] = 0; // 如果b.length不足4,则将高位补0
    }
    // 注意此处和byte数组转换成int的区别在于，下面的转换中要将先将数组中的元素转换成long型再做移位操作，
    // 若直接做移位操作将得不到正确结果，因为Java默认操作数字时，若不加声明会将数字作为int型来对待，此处必须注意。
    long v0 = (long) (a[0] & 0xff) << 56; // &0xff将byte值无差异转成int,避免Java自动类型提升后,会保留高位的符号位
    long v1 = (long) (a[1] & 0xff) << 48;
    long v2 = (long) (a[2] & 0xff) << 40;
    long v3 = (long) (a[3] & 0xff) << 32;
```

```
long v4 = (long) (a[4] & 0xff) << 24;
long v5 = (long) (a[5] & 0xff) << 16;
long v6 = (long) (a[6] & 0xff) << 8;
long v7 = (long) (a[7] & 0xff);
return v0 + v1 + v2 + v3 + v4 + v5 + v6 + v7;
}
```

步骤六：题目的扩散

如果您使用自己的 IM 系统下发题目，请忽略这一部分，如果想使用腾讯云 IM 服务发题，请对接如下几步：

• 1. 开通 IM 通讯服务

开通腾讯云 [云通信](#) 服务。

• 2. 配置 IM 通讯服务

按照文档进行初始化[配置](#)，集成模式请务必选择**独立模式**。

• 3. 使用 [REST API](#) 创建一个 BChatRoom 用于发题

腾讯云 IM 的 [REST API](#) 是专门提供给服务端接入用的，创建群组这个操作一般是由您的服务器触发，所以适合使用 REST API 方案进行接入。

BChatRoom 这种房间类型非常适合用来发题，因为它原本就是用来做系统通知的，所以消息达到率高，消息可靠性好。

创建群组使用 [v4/group_open_http_svc/create_group](#) 可以实现，测试方法见 [SDK](#) 开发包中的 [IM后台RESTAPI使用方法.pdf](#) - 步骤3。

• 4. 使用 [REST API](#) 创建一个 AVChatRoom 用于弹幕

AVChatRoom 这种房间类型非常适合用来做聊天室的弹幕消息，有比较严格的脏字过滤和频率限制逻辑，专门为大型聊天室场景优化。

创建群组使用 [v4/group_open_http_svc/create_group](#) 可以实现，测试方法见 [SDK](#) 开发包中的 [IM后台RESTAPI使用方法.pdf](#) - 步骤4。

AVChatroom中默认的频率控制为40条/s，如果您需要调整限制请联系我们，因为消息越多，带宽费用越高。

• 5. 使用 [REST API](#) 在 BChatRoom 里发送题目广播

消息发送可以使用 [v4/group_open_http_svc/send_group_msg](#)，测试方法见 [SDK](#) 开发包中的 [IM后台RESTAPI使用方法.pdf](#) - 步骤5。

步骤七：题目接收&弹幕收发

客户端使用 IM SDK 完成消息的接收和弹幕消息的收发，这里可以按照如下步骤对接

1. 集成精简版 IMSDK

精简版 IMSDK 位于 [SDK开发包](#) 中，zip包中的“接入指引-IMSDK.pdf”有详细的说明。

2. 参考源码完成对接

SDK开发包中有一个叫做 **AnswerPlayIMCenter** 的源码文件，封装了对于 IMSDK 的简单调用，相当于对于接入文档的代码示例，您可以参考一下，如下是这个类的成员函数说明：

成员函数	作用
initIMCenter	用于初始化，需要填写您在腾讯云的 IM 服务相关信息。
loginIMUser	用于登录，您可以把 imSDK 当成一个无界面的 QQ，用QQ收发消息肯定要登录，只是将QQ号和登录密码换成了您的Userid和您的服务器签发的UserSig。
joinIMGroup	用于加入在步骤六中由您的后台服务器通过 REST API 创建的 BChatRoom 和 AVChatRoom。
sendChatMessage	用来发送弹幕评论消息
onRecvChatMessage	用于接收来自 AVChatRoom 的弹幕消息，注意做一下渲染的频率限制， 不要收到一条弹幕消息就刷一次屏幕 ，手机性能可能扛不住，很多客户都是在这里疏忽了，因为测试期间消息不多不易发现。
onRecvIssueMessage	用于接收来自 BChatRoom 的题目消息，按照方案二的设计，每道题目里面都有它应该显示的 NTP 国际时间。

3. 题目的显示时机

onRecvIssueMessage 收到的题目不要立刻显示出来，要等来自播放器的 GET_MESSAGE 回调，如果回调的时间 $\geq$ 题目的 NTP 国际时间，就可以把指定的题目显示出来了。

4. UserSig怎么算？

UserSig 是 loginIMUser 时需要的一个重要信息，它的作用等同于登录QQ用的密码，只是这个密码由您的服务器为您的用户签发；这里使用的密码签发方案是基于 RSA 非对称加密实现的，所以安全性非常高。

UserSig 一般是由您的服务器签发的，签发方案可以参考：[TLS 后台 API](#)，不过为了能够让您快速调试起来，我们也提供了一个 Windows 的签发[小工具](#)，您可以在后台同学加入项目之前就可以先把终端逻辑调试起来。

注意：小工具里的公私钥都只能用来测试，您可千万别当真的用了。

步骤八：开发答题系统

由于腾讯云 PAAS 的定位，所以对于跟客户业务绑定比较紧密的答题和支付系统，我们就不涉及了，需要您来参与开发。

这里普遍采用的方案是：将客户的答案以 HTTP(S) 请求方式汇总到答题服务器上，只是实现过程需要注意解决瞬间高并发的请求压力。

有客户可能会问，IM系统是否适合用来答题，这里目前看是不适合的，因为 IM 系统的主要能力在于消息扩散，而答题的主要目标是信息的收拢。

步骤九：答题结果显示

一般题目出来一段时间以后，就会进入闭题状态。这时，答题系统会将结果进行统计汇总，并将汇总结果下发给观众，结果下发可以继续使用 **步骤六** 中的题目扩散通道。