

# 直播 SDK SDK 下载 产品文档





#### 【版权声明】

#### ©2013-2022 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

SDK 下载

SDK 下载

SDK 集成

iOS

Android

功能说明

RTC 推流优势



# SDK 下载 SDK 下载

最近更新时间: 2022-09-15 10:23:07

直播 SDK 提供专业版 SDK, 了解专业版 SDK 功能及其 License 使用可参见 功能说明。

### 专业版(International)

专业版集合了包含移动直播在内的多个音视频相关的核心功能,这包括实时音视频 SDK、直播推拉流和基础美颜能力。

所属平台	ZIP 包	64位支持	安装包增量	安装包瘦身
iOS	DOWNLOAD	支持	5.69M (arm64)	DOC
Android	DOWNLOAD	支持	9.15M (armv7) 10.4M (arm64)	DOC

### 说明:

根据您具体使用的服务, 使用专业版需先购买对应产品授权, 您可按需选购:

- 使用其中的移动直播请购买 直播 SDK 专业版 License。
- 使用其中的实时音视频请购买实时音视频套餐包。
- 在使用 MLVB SDK时,您需要先设置相关环境。相关示例代码如下:

#### Android:

V2TXLivePremier.setEnvironment("GDPR");

#### iOS:

[V2TXLivePremier setEnvironment:@"GDPR"];



# SDK 集成

### iOS

最近更新时间: 2022-09-14 17:29:41

本文主要介绍如何快速地将腾讯云视立方·移动直播 LiteAVSDK(iOS)集成到您的项目中,按照如下步骤进行配置,就可以完成 SDK 的集成工作。

### 开发环境要求

- Xcode 9.0+
   <sub>o</sub>
- iOS 9.0 以上的 iPhone 或者 iPad 真机。
- 项目已配置有效的开发者签名。

### 集成 LiteAVSDK

您可以选择使用 CocoaPods 自动加载的方式,或者先下载 SDK,再将其导入到您当前的工程项目中。

#### CocoaPods

#### 1. 安装 CocoaPods

在终端窗口中输入如下命令(需要提前在 Mac 中安装 Ruby 环境):

```
sudo gem install cocoapods
```

#### 2. 创建 Podfile 文件

进入项目所在路径,输入以下命令行之后项目路径下会出现一个 Podfile 文件。

pod init

#### 3. 编辑 Podfile 文件

编辑 Podfile 文件,有如下有两种设置方式:

• 方式一:使用腾讯云 LiteAVSDK 的 podspec 文件路径。

```
platform :ios, '9.0'
```



```
target 'App' do
pod 'TXLiteAVSDK_Professional', :podspec => 'https://liteav.sdk.qcloud.com/po
d/liteavsdkspec/TXLiteAVSDK_Professional.podspec'
end
```

• 方式二:使用 CocoaPod 官方源,支持选择版本号。

```
platform :ios, '9.0'
source 'https://github.com/CocoaPods/Specs.git'

target 'App' do
pod 'TXLiteAVSDK_Professional'
end
```

#### 4. 更新并安装 SDK

在终端窗口中输入如下命令以更新本地库文件,并安装 LiteAVSDK:

```
pod install
```

或使用以下命令更新本地库版本:

```
pod update
```

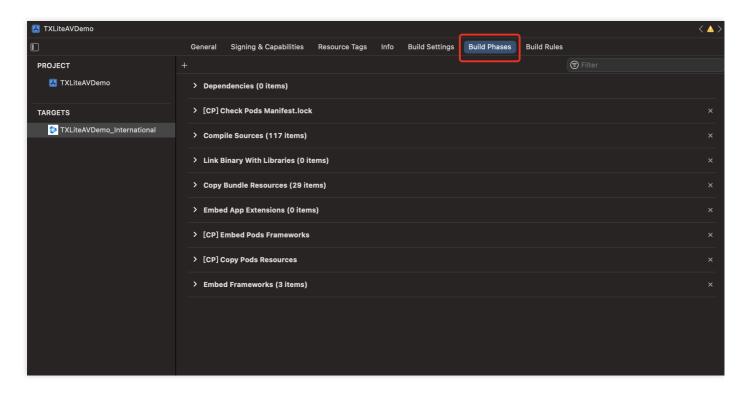
pod 命令执行完后,会生成集成了 SDK 的 .xcworkspace 后缀的工程文件,双击打开即可。

### 手动集成

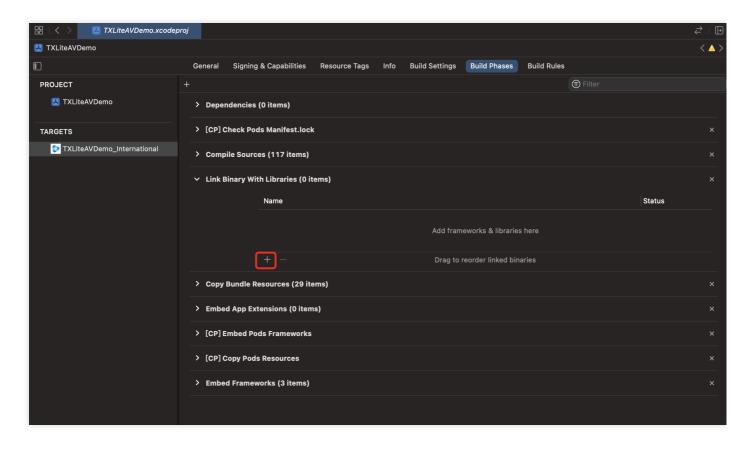
1. 下载 LiveAVSDK ,下载完成后进行解压。



2. 打开您的 Xcode 工程项目,选择要运行的 target,选中 Build Phases 项。



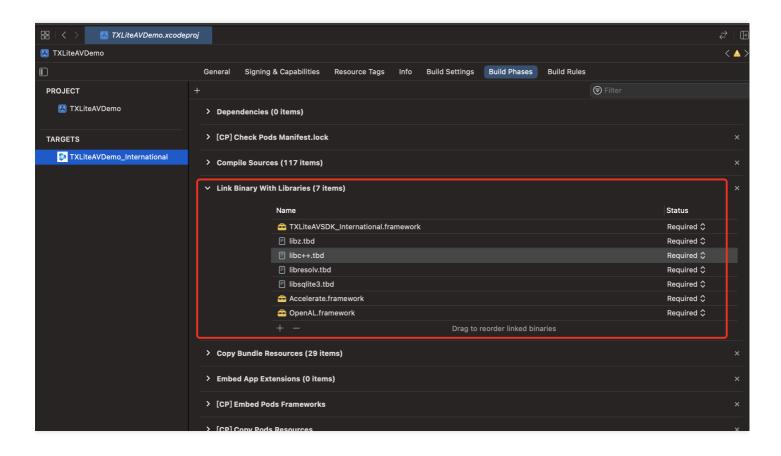
3. 单击 Link Binary with Libraries 项展开,单击底下的【+】添加依赖库。



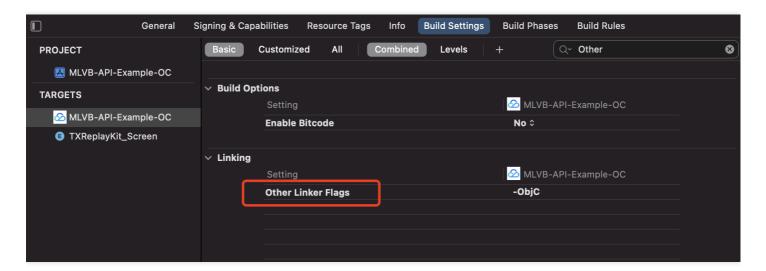
4. 依次添加所下载的 TXLiteAVSDK\_Professional.framework 及其所需依赖库:



libz.tbd
libc++.tbd
libresolv.tbd
libsqlite3.tbd
Accelerate.framework
OpenAL.framework



5. 选中 Build Settings 项,搜索 Other Linker Flags 。添加 -ObjC 。





### 授权摄像头和麦克风使用权限

使用 SDK 的音视频功能,需要授权麦克风和摄像头的使用权限。在 App 的 Info.plist 中添加以下两项,分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

- Privacy Microphone Usage Description, 并填入麦克风使用目的提示语。
- Privacy Camera Usage Description, 并填入摄像头使用目的提示语。



### 在工程中引入 SDK

项目代码中使用 SDK 有两种方式:

• 方式一: 在项目需要使用 SDK API 的文件里,添加模块引用。

```
@import TXLiteAVSDK_Professional;
```

• 方式二:在项目需要使用 SDK API 的文件里,引入具体的头文件。

```
#import "TXLiteAVSDK_Professional/TXLiteAVSDK.h"
```



### 给 SDK 配置 License 授权

登录云直播控制台,在左侧菜单中选择 **直播 SDK > License 管理\*\*,单击 \*\*Get License** 获取测试用 License(详细操作请参见 [申请测试版 License](https://intl.cloud.tencent.com/document/product/1071/38546)。您会获得两个字符串:一个字符串是 LicenseURL,另一个字符串是解密 Key。

在您的 App 调用 LiteAVSDK 的相关功能之前(建议在 – [AppDelegate application:didFinishLaunchingWithOptions:] 中)进行如下设置:

```
@import TXLiteAVSDK_Professional;
@implementation AppDelegate
- (BOOL)application: (UIApplication *)application didFinishLaunchingWithOptions: (N
SDictionary *)launchOptions {
NSString * const licenceURL = @"<获取到的licenseUrl>";
NSString * const licenceKey = @"<获取到的key>";

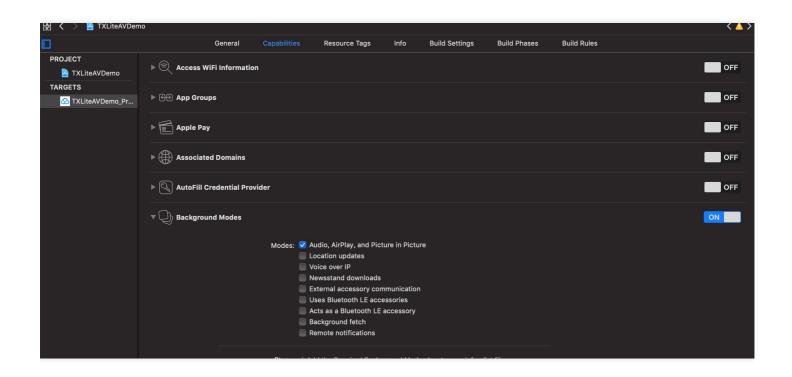
//TXLiveBase 位于 "TXLiveBase.h" 头文件中
[TXLiveBase setLicenceURL:licenceURL key:licenceKey];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
@end
```

### 常见问题

#### LiteAVSDK 是否支持后台运行?

支持,如需要进入后台仍然运行相关功能,可选中当前工程项目,在 Capabilities 下设置 Background Modes 为 ON,并勾选 Audio,AirPlay and Picture in Picture,如下图所示:







### **Android**

最近更新时间: 2022-09-14 17:31:28

本文主要介绍如何快速地将腾讯云 LiteAVSDK(Android)集成到您的项目中,按照如下步骤进行配置,就可以完成 SDK 的集成工作。

### 开发环境要求

- Android Studio 2.0+0
- Android 4.1 (SDK API 16) 及以上系统。

### 集成 SDK (aar)

您可以选择使用 Gradle 自动加载的方式,或者手动下载 aar 再将其导入到您当前的工程项目中。

### 方法一:自动加载 (aar)

因为 jcenter 已经下线,您可以通过在 gradle 配置 mavenCentral 库,自动下载更新 LiteAVSDK。只需要用 Android Studio 打开需要集成 SDK 的工程,然后通过简单的四个步骤修改 build.gradle 文件,就可以完成 SDK 集成:

```
Simple of the property of the
```

1. 打开 app 下的 build.gradle。



2. 在 dependencies 中添加 LiteAVSDK 的依赖。

```
dependencies {
  implementation 'com.tencent.liteav:LiteAVSDK_Professional:latest.release'
  }

或
```

```
dependencies {
  implementation 'com.tencent.liteav:LiteAVSDK_Professional:latest.release@aar'
}
```

3. 在 defaultConfig 中,指定 App 使用的 CPU 架构(目前 LiteAVSDK 支持 armeabi、 armeabi-v7a 和 arm64-v8a)。

```
defaultConfig {
  ndk {
  abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
  }
}
```

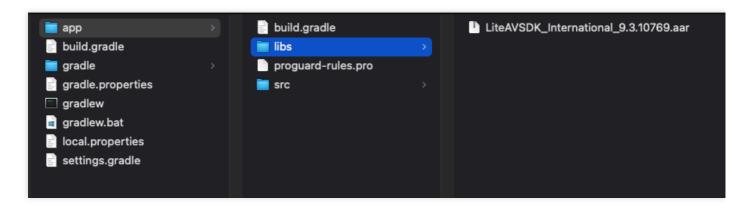


4. 单击 Sync Now 按钮同步 SDK,如果您的网络连接 mavenCentral 没有问题,很快 SDK 就会自动下载集成到工程里。

#### 方法二:手动下载 (aar)

如果您的网络连接 mavenCentral 有问题,也可以手动下载 SDK 集成到工程里:

- 1. 下载 LiteAVSDK ,下载完成后进行解压。
- 2. 将下载文件解压之后 SDK 目录下的 aar 文件拷贝到工程的 app/libs 目录下:





3. 在工程根目录下的 build.gradle 中,添加 flatDir,指定本地仓库路径。

```
LiteAVSDKDemo × ap
                                            // Top-level build file where you can add configuration options common to all sub-projects/modules.
  gradle
.idea
                                            buildscript {
▶ app▼ mgradle
                                                  repositories {
                                                        google()
  ▶ m wrapper

d .gitignore
                                                        jcenter()
  build.gradle
  gradle.properties
gradlew.bat
LiteAVSDKDemo.iml
settings.gradle
Scratches and Consoles
                                                  repositories {
                                                       google()
                                                       flatDir {
dirs 'libs
                                            task clean(type: Delete) {
    delete rootProject.buildDir
```

4. 添加 LiteAVSDK 依赖,在 app/build.gradle 中,添加引用 aar 包的代码。

```
| Part |
```

```
implementation(name:'LiteAVSDK_Professional_8.7.10102', ext:'aar')
```

5. 在 app/build.gradle 的 defaultConfig 中,指定 App 使用的 CPU 架构(目前 LiteAVSDK 支持 armeabi、 armeabi-v7a 和 arm64-v8a)。



```
defaultConfig {
ndk {
  abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
  }
}
```

6. 单击 Sync Now 按钮同步 SDK, 完成 LiteAVSDK 的集成工作。

### 集成 SDK (jar)

如果您不想集成 aar 库,也可以通过导入 jar 和 so 库的方式集成 LiteAVSDK:

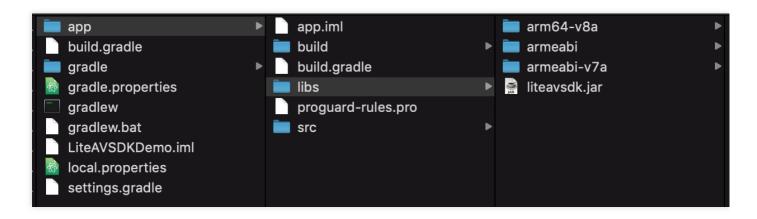
1. 下载 LiteAVSDK, 下载完成后进行解压。在 SDK 目录下找到 LiteAVSDK\_Professional\_xxx.zip (其中 xxx 为 LiteAVSDK 的版本号):



解压后得到 libs 目录, 里面主要包含 jar 文件和 so 文件夹, 文件清单如下:



2. 将解压得到的 jar文件和 armeabi、armeabi-v7a、arm64-v8a 文件夹拷贝到 app/libs 目录下。





3. 在 app/build.gradle 中,添加引用 jar 库的代码。

```
gradle
.idea
                                                     apply plugin: 'com.android.application
  app

build
                                                     android {
compileSdkVersion 28
  ▶ 🖿 libs
▶ 🖿 src
                                                            defaultConfig {
     gitignore app.iml
  w build.gradle

gradle proguard_rules.pro

gradle
                                                                 targetSdkVersion 28
                                                                 versionCode 1
                                                                 versionName "1.0"
  d gitignore

⇒ build.gradle

gradle.properties

gradlew

gradlew

gradlew
  LiteAVSDKDemo.iml
                                                           buildTypes {
settings gradle
                                                                 release {
Scratches and Consoles
                                                                       proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
                                                                       jniLibs.srcDirs = ['libs']
                                                     dependencies {
                                                           implementation fileTree(dir: 'libs', include: ['*.jar'])
                                                            implementation 'com.android.support:appcompat-v7:28.0.0'
                                                          implementation 'com.android.support.constraint:constraint-layout:1.1.3'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
```

```
dependencies{
implementation fileTree(dir:'libs',include:['*.jar'])
}
```



4. 在工程根目录下的 build.gradle 中,添加 flatDir,指定本地仓库路径。

```
No Devices ▼ ▶ ☆ 등 ಈ ೄ ⋒ 등 ■ № ◘ • □ ► □ Q
LiteAVSDKDemo > 2 build.gradle
                                          // Top-level build file where you can add configuration options common to all sub-pr
   LiteAVSDKDemo ~/AndroidStudioProj
   ▶ ■ .gradle▶ ■ .idea
                                             repositories {
   🔻 📭 арр
     ▶ Ilibs
       🚜 .gitignore
                                                  classpath "com.android.tools.build:gradle:4.1.2"
       🔊 build.gradle
        🛔 proguard-rules.pro
   ▶ m gradle
     륂 .gitignore
   🚮 gradle.properties
      ■ gradlew
                                              repositories {
      \rm gradlew.bat
                                                  google()
      🚮 local.properties
                                                  mavenCentral()
      settings.gradle
 ► III External Libraries
   Scratches and Consoles
                                          task clean(type: Delete) {
                                           🍦 delete rootProject.buildDir
```



5. 在 app/build.gradle 中,添加引用 so 库的代码。

```
gradl
idea
app
                                               apply plugin: 'com.android.application'
  build libs
    src
gitignore
app.iml
                                                     defaultConfig {
                                                          applicationId "com.tencent.liteavsdkdemo"
                                                          minSdkVersion 21
    🚙 build.gradle
                                                          targetSdkVersion 28
  proguard_rules.pro
  wrapper gitignore
 angliore

in gradle properties

gradlew
gradlew
gradlew
LiteAVSDKDemo.iml
                                                     buildTypes {
settings.gradle
                                                          release {
Scratches and Consoles
                                                                proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
                                                    sourceSets {
                                                                jniLibs.srcDirs = ['libs']
                                               dependencies {
                                                    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
testImplementation 'junit:junit:4.12'
                                                    androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
```

6. 在 app/build.gradle 的 defaultConfig 中,指定 App 使用的 CPU 架构(目前 LiteAVSDK 支持 armeabi、 armeabi-v7a 和 arm64-v8a)。

```
defaultConfig {
  ndk {
  abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
  }
}
```

7. 单击 Sync Now 按钮同步 SDK, 完成 LiteAVSDK 的集成工作。

### 配置 App 打包参数



```
PituDemo [E:\MvStudioWorkspace2\PituDemo] - app - Android Studio
 <u> Elle Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help</u>
📮 PituDemo 🕽 📭 app 🕽 😉 build.gradle
                              🔻 🤀 🗦 | 🕸 👫 🔭 🔞 CameraPusherActivity.java × 🚺 app 🗴 🔞 PusherSettingFragment.java × 🔞 MainActivity.java × 🖁 activity_main.xml ×
   ▼ 📙 PituDemo E:\MyStudioWorkspace2\Pitu
      ▼ 📭 app
        build
                                                                     ndk {
         ▶ src
            # .gitignore
          🕝 build.gradle
           proguard-rules.pro
      ▶ In player
      ▶ In trtc
      ▶ In videojoiner
                                                                     doNotStrip "*/armeabi-v7a/libYTCommon.so"
doNotStrip "*/x86/libYTCommon.so"
         \rm gitignore
         gradlew
```

```
packagingOptions {
pickFirst '**/libc++_shared.so'
doNotStrip "*/armeabi/libYTCommon.so"
doNotStrip "*/armeabi-v7a/libYTCommon.so"
doNotStrip "*/x86/libYTCommon.so"
doNotStrip "*/arm64-v8a/libYTCommon.so"
}
```

### 配置 App 权限

在 AndroidManifest.xml 中配置 App 的权限, LiteAVSDK 需要以下权限:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" /></uses-permission android:name="android.permission.READ_PHONE_STATE" />
```



```
<uses-feature android:name="android.hardware.Camera"/>
<uses-feature android:name="android.hardware.camera.autofocus" />
```

### 配置 License 授权

登录云直播控制台,在左侧菜单中选择 **直播 SDK > License 管理\*\*,单击 \*\*Get License** 获取测试用 License(详细操作请参见 [申请测试版 License](https://intl.cloud.tencent.com/document/product/1071/38546)。您会获得两个字符串:一个字符串是 LicenseURL,另一个字符串是解密 Key。

在您的 App 调用企业版 SDK 相关功能之前(建议在 Application类中)进行如下设置:

```
public class MApplication extends Application {

@Override
public void onCreate() {
    super.onCreate();
    String licenceURL = ""; // 获取到的 licence url
    String licenceKey = ""; // 获取到的 licence key
TXLiveBase.getInstance().setLicence(this, licenceURL, licenceKey);
    }
}
```

### 设置混淆规则

在 proguard-rules.pro 文件中,将 LiteAVSDK 相关类加入不混淆名单:

```
-keep class com.tencent.** { *; }
```



# 功能说明

最近更新时间: 2022-09-14 17:27:35

移动直播目前为国际站用户特别提供**专业版 SDK**,可通过移动直播专业版 License 使用相应的直播功能,包括直播推流、直播播放和基础美颜(美白、磨皮等)等。

#### SDK 与授权 License

您可以通过申请测试的移动直播专业版 License,或购买正式的移动直播专业版 License 对移动直播专业版 SDK 进行使用。

### SDK 功能列表

功能模块	功能项	功能简介	专业版 International
界面	自定义 UI	开发者自定义 UI。小直播 App 提供了一套完整的 UI 交互源码,可复用或自定义。	✓
RTMP 推 直播推流		用于实现主播端的手机推流功能(秀场直播)。	✓
	录屏推流	用于实现主播端的屏幕推流功能(游戏直播)。	<b>✓</b>
直播播放	RTMP 播 放	用于实现 RTMP 协议的直播播放功能。	✓
	FLV 播放	用于实现 HTTP + FLV 协议的直播播放功能。	<b>✓</b>
	HLS 播放	用于实现 HLS(m3u8)协议的直播播放功能。	<b>✓</b>
	WebRTC 播放	用于实现快直播(LEB)的直播播放功能。	✓
点播播放	点播播放	用于实现视频点播回放功能。	<b>✓</b>
直播连麦	连麦互动	用于实现主播与观众之间的1vn视频连麦互动。	/
	主播 PK	用于实现主播与主播之间的1v1视频 PK。	<b>✓</b>
采集拍摄	屏比	支持16:9, 4:3, 1:1多种屏比拍摄。	<b>✓</b>
	清晰度	支持标清、高清及超清拍摄,支持自定义码率、帧率及 gop。	<b>✓</b>
	拍摄控制	拍摄前后摄像头切换和灯光的控制。	1



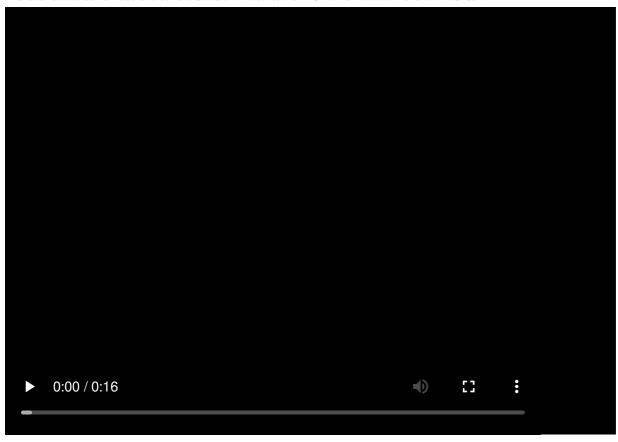
功能模块	功能项	功能简介	专业版 International
	水印	拍摄支持添加水印。	✓
	焦距	拍摄支持调节焦距。	<b>✓</b>
	对焦模式	支持手动对焦和自动对焦。	✓
	拍照	支持拍摄照片。	✓
	背景音乐	拍摄前可以选择本地的 MP3 作为背景音。	<b>✓</b>
	变声和混 响	拍摄前对录制的声音变声(如萝莉、大叔)和混响效果(如KTV、会堂)。	✓
	滤镜	支持自定义滤镜及设置滤镜程度。	✓
	基础美颜	拍摄设置面部磨皮、美白、红润并调节强度。	✓
	高级美颜	拍摄设置大眼、瘦脸、V 脸、下巴调整、短脸、小鼻效果, 并支持调节强度。	×
	动效贴纸	定位五官位置,然后添加变形、覆盖贴纸挂件等效果。(通过素材提供)	×
	AI 抠图	识别出背景轮廓, 把背景抠除, 替换成素材视频的元素。(通过素材提供)	×
	绿幕抠像	将画面中的绿色元素(如纯绿背景)抠除,替换成其他的元素。	×



# RTC 推流优势

最近更新时间: 2022-07-20 10:50:17

为了提高在弱网下的推流效果,直播 SDK 在传统的 RTMP 协议推流的基础上增加了 RTC 协议的推流方式。本文档主要介绍不同档位弱网条件下两种协议的数据对比。效果可以参考下面的视频:



#### 说明:

推流教程请参见摄像头推流。

### 正常网络及弱网环境下的效果质量

### 测试场景

主播推流,观众通过 CDN 观看。在主播端进行各种弱网限制,观察观众播放的效果质量。本文档弱网环境只针对上行,下行网络为无损状态。

#### 参数配置

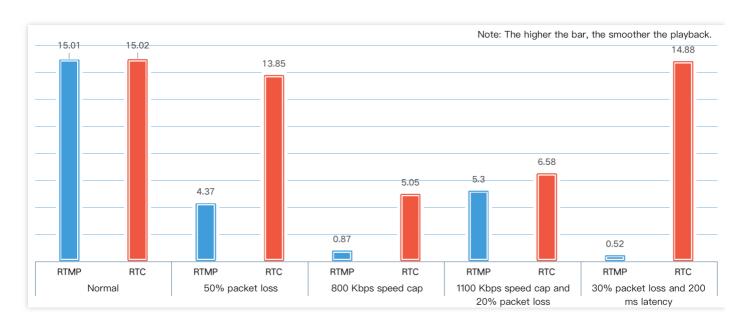


为了避免受不同源的影响,使用 RTMP 和 RTC 推流时都固定使用 V2TXLivePusher 推同一个本地视频。视频参数:

参数类型	配置信息
分辨率	720 × 1280
码率	1800 kbps
帧率	15

### 几种弱网场景下关键指标的对比

#### • 帧率

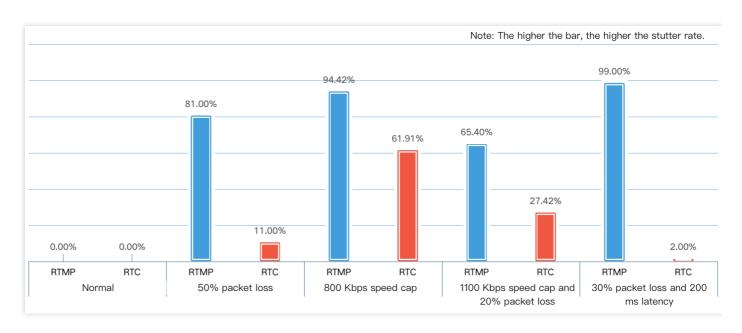


说明:

具体参数含义请参见 附录:网络参数说明。



#### 卡顿率



说明:

具体参数含义请参见 附录:网络参数说明。

### 附录:网络参数说明

参数	说明
帧率	每秒钟渲染帧数
丢包率	50% 代表发送了10个数据包中会丢失5个
时延	200ms 时延代表 SDK 发送的包,会经过 200ms 后才被网络发送出去
限速	800kbps 代表每秒钟最多发送 800kb 的数据
卡顿率	渲染间隔大于 200ms 表示卡顿,卡顿率等于卡顿时间总和除以总播放时长