# 实时音视频

# 快速入门

# 产品文档

腾讯云

# 文档目录

- 快速入门
  - 快速集成 SDK
    - 快速集成(iOS)
    - 快速集成(Android)
    - 快速集成(Mac)
    - 快速集成(Windows)
    - 快速集成(Web)
    - 快速集成(Electron)
    - 快速集成(Flutter)
    - 快速集成(QT)
    - 快速集成(Unity)
    - 快速集成(React Native)
    - 快速集成(Unreal Engine)
  - 快速跑通示例
    - iOS&Mac
    - Android
    - Windows
    - Web
    - Electron
    - Flutter
    - Unity
    - React Native
    - Unreal Engine
  - 新手常见问题

# 快速入门
# 快速集成 SDK
# 快速集成(iOS)

最近更新时间：2022-04-02 16:27:02

本文主要介绍如何快速地将腾讯云 TRTC SDK（iOS）集成到您的项目中，只要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+。
- 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 TRTC SDK

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

```
platform :ios, '8.0'
target 'App' do
pod 'TXLiteAVSDK_TRTC', :podspec => 'http://pod-1252463788.cosgz.myqcloud.com/liteavsdkspec/TXLit
```

```
eAVSDK_TRTC.podspec'
end
```

您也可以使用 CocoaPod 官方源，但下载速度可能较慢：

```
platform :ios, '8.0'
source 'https://github.com/CocoaPods/Specs.git'

target 'App' do
pod 'TXLiteAVSDK_TRTC'
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 TRTC SDK：

```
pod install
```

或使用以下命令更新本地库版本：

```
pod update
```

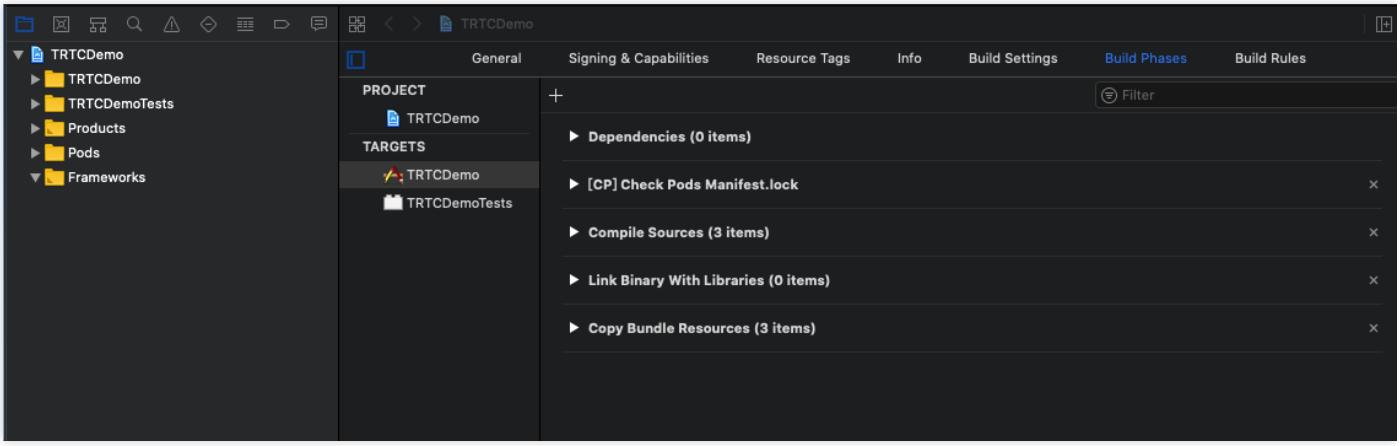
pod 命令执行完后，会生成集成了 SDK 的 .xcworkspace 后缀的工程文件，双击打开即可。

> 说明：
> 需要手动添加所需系统依赖库 **Accelerate.framework**。

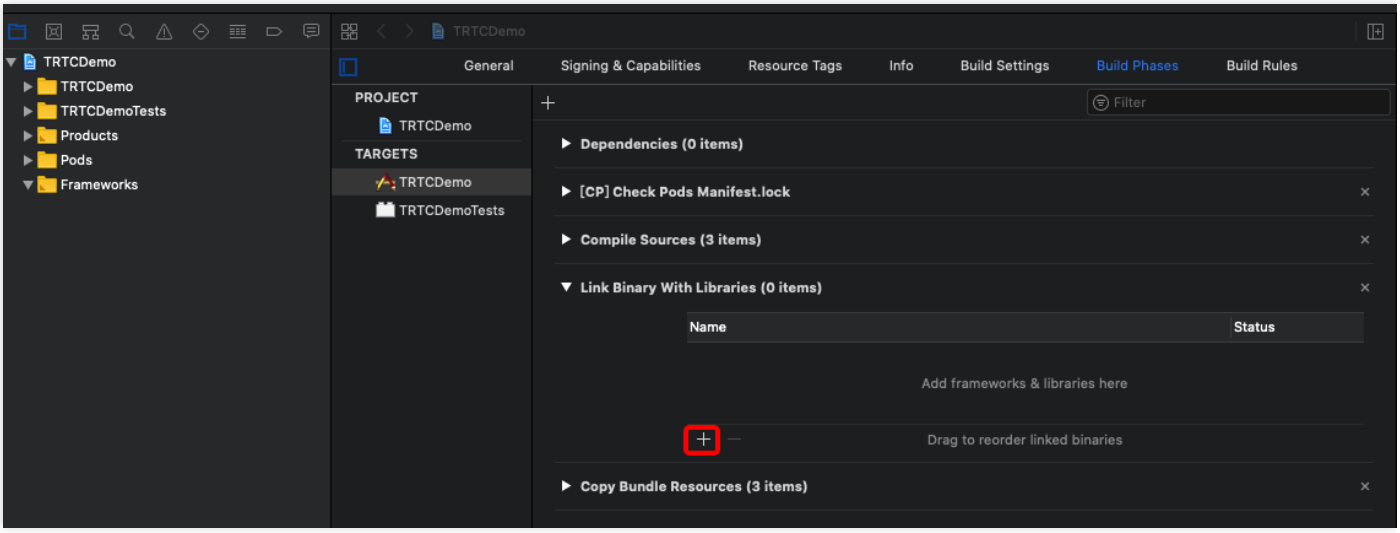
### 手动集成

1. 下载 TRTC - SDK ，下载完成后进行解压。

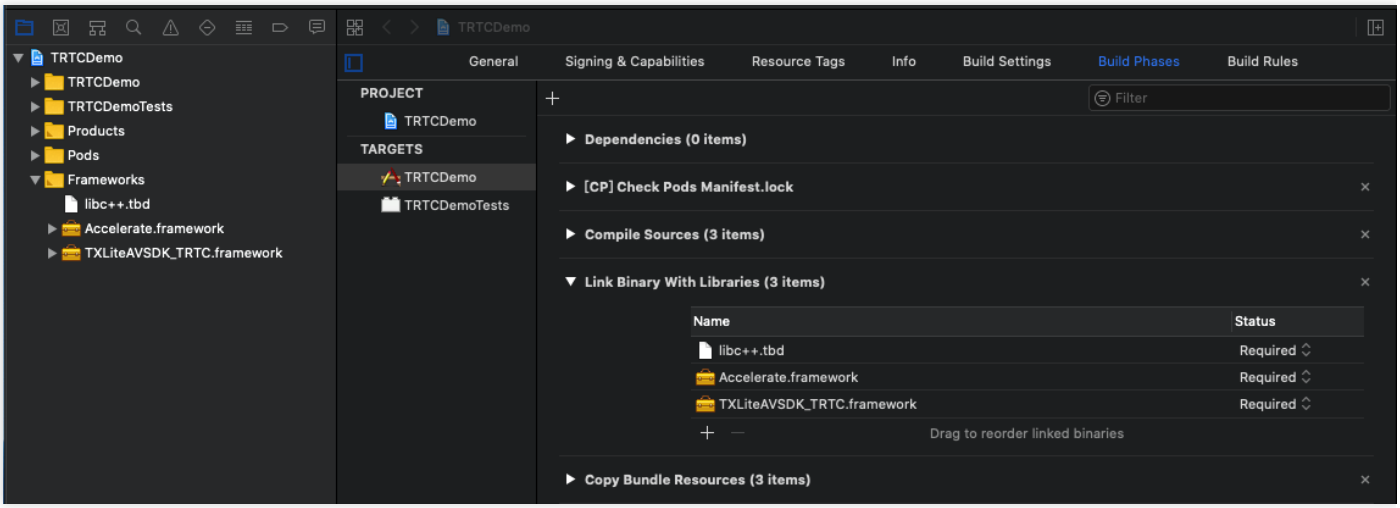
2. 打开您的 Xcode 工程项目，选择要运行的 target，选中 **Build Phases** 项。



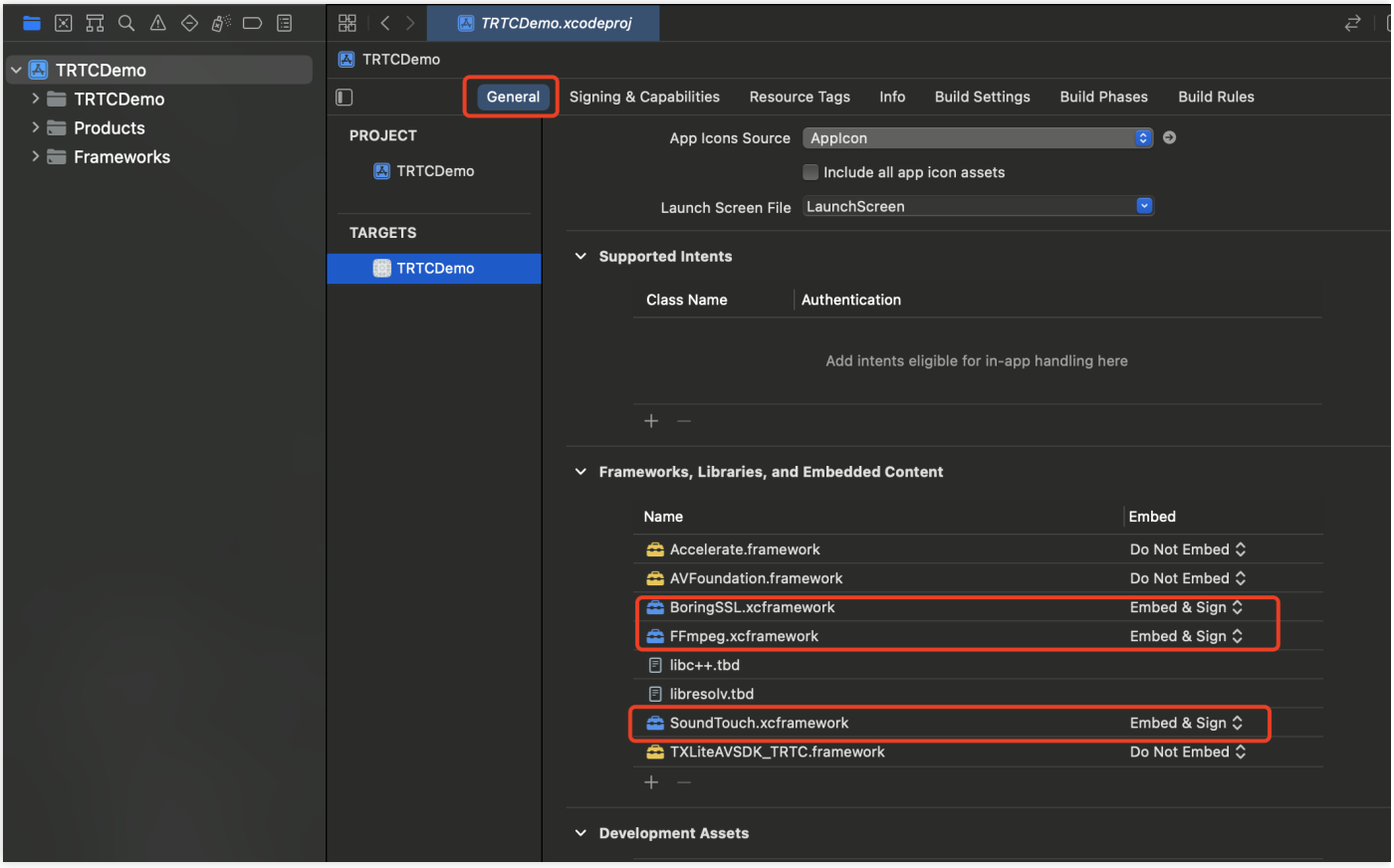

3. 单击 **Link Binary with Libraries** 项展开，单击底下的“+”号图标去添加依赖库。




4. 依次添加所下载的 TRTC SDK Framework 及其所需依赖库 **libc++.tbd** 、**Accelerate.framework** 和 **libresolv.tbd**、**AVFoundation.framework**。



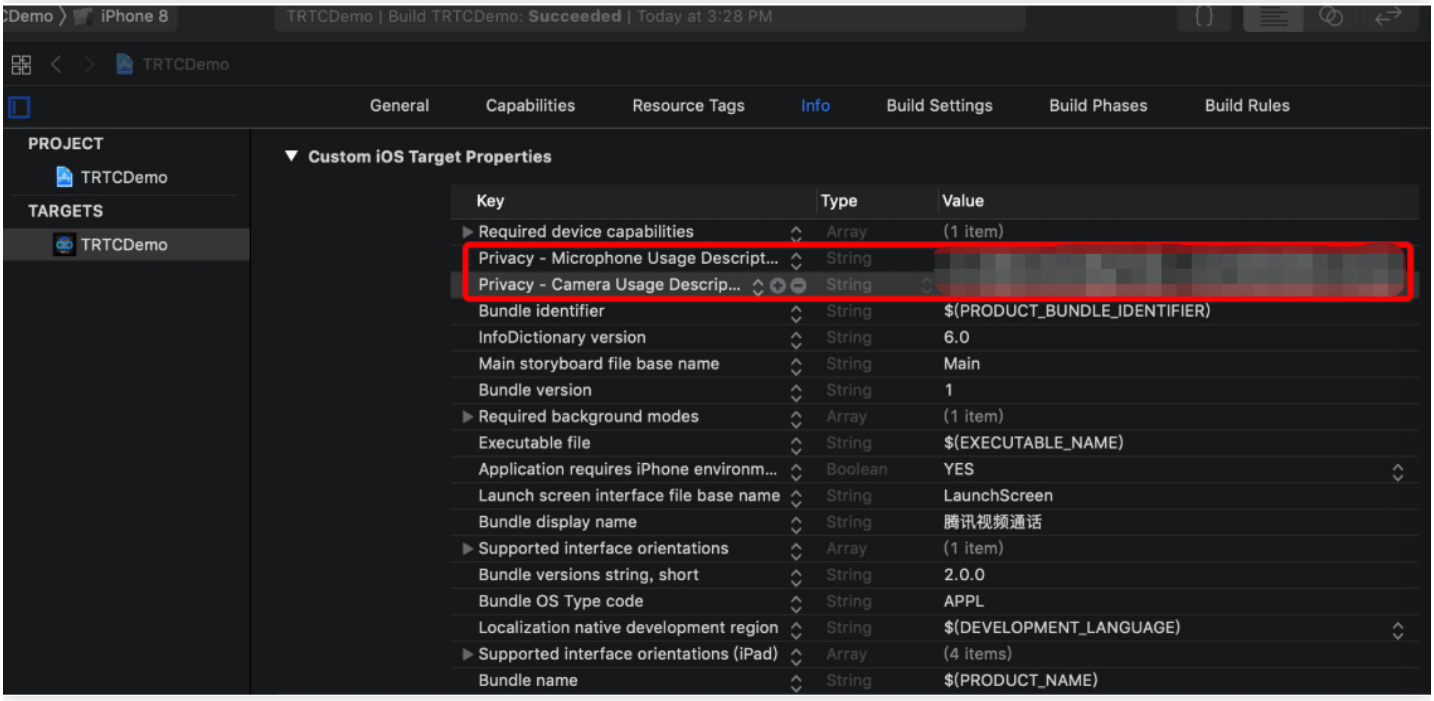

5. **TRTC SDK 9.5.11234 及以上版本** 需要添加动态库依赖。
   单击 **General**，选择 **Frameworks,Libraries,and Embedded Content**，单击底下的“+”号图标依次添加 TXLiteAVSDK_TRTC.framework 所需要动态库 **BoringSSL.xcframework**、**FFmpeg.xcframework**、**SoundTouch.xcframework**，选择 **Embed & Sign**。




# 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。




# 引用 TRTC SDK

TRTC SDK 支持两种调用方式，您可以任选一种

### 方案一：通过 Objective-C 或 Swift 接口引用 TRTC SDK

在 Objective-C 或 Swift 代码中使用 SDK 有两种方式：

- **模块引用**：在项目需要使用 SDK API 的文件里，添加模块引用。

```
@import TXLiteAVSDK_TRTC;
```

- **头文件引用**：在项目需要使用 SDK API 的文件里，引入具体的头文件。

```
#import TXLiteAVSDK_TRTC/TRTCCloud.h
```

### 方案二：通过 C++ 接口引用 TRTC SDK

1. **引用头文件**：如果您要使用 C++ 接口来开发 iOS 应用，请引用 `TXLiteAVSDK_TRTC.framework/Headers/cpp_interface` 目录下的头文件

```
#include TXLiteAVSDK_TRTC/cpp_interface/ITRTCCloud.h
```

2. **使用命名空间**：C++ 全平台接口的方法、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间

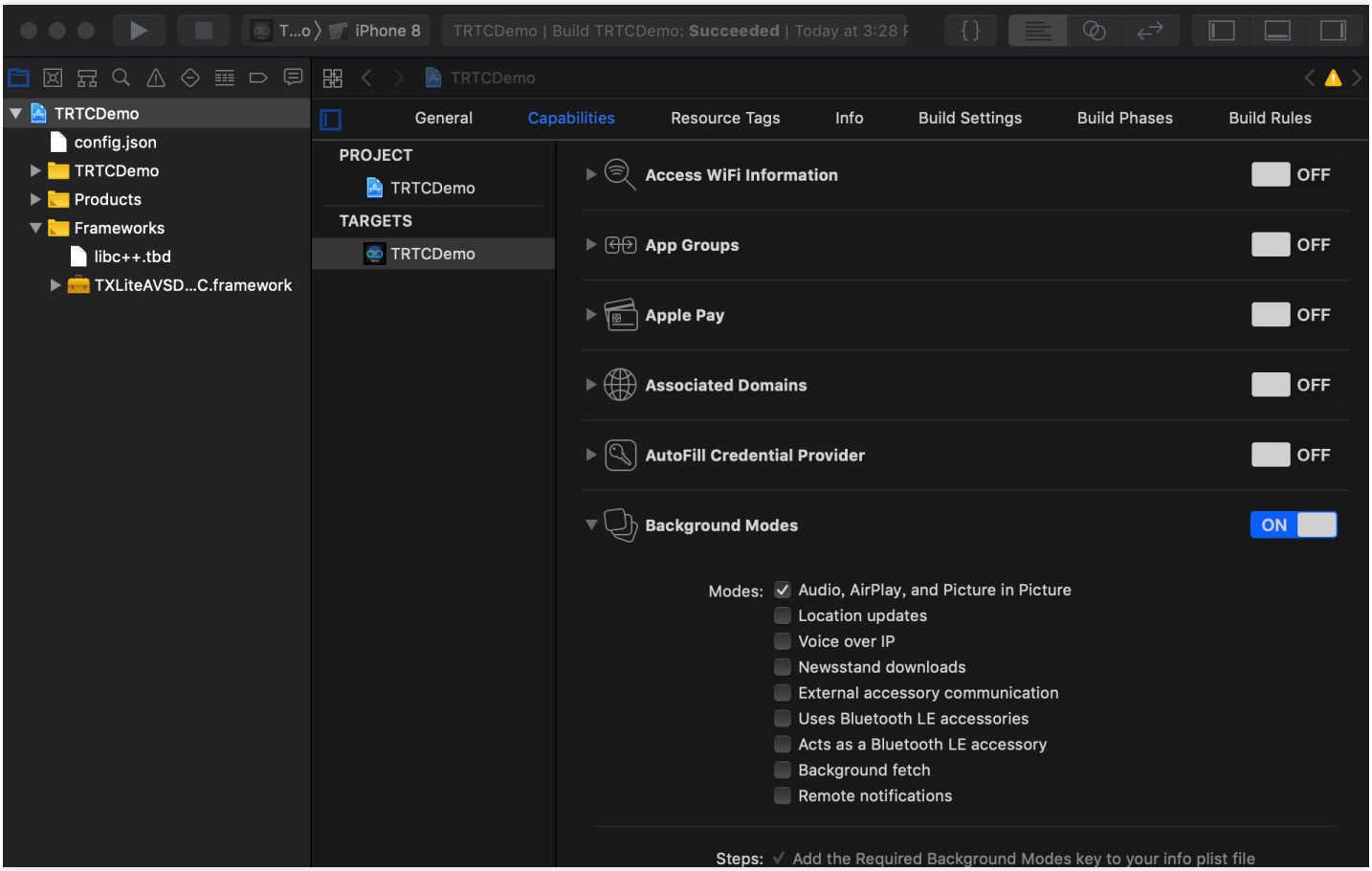
```
using namespace trtc;
```

> 说明：
> 对于 C++ 接口的使用方式，请参见 全平台（C++）API 概览。

## 常见问题

### TRTC SDK 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** 下的设置 **Background Modes** 打开为 **ON**，并勾选 **Audio，AirPlay and Picture in Picture**，如下图所示：


iPhone 8
TRTCDemo | Build TRTCDemo: Succeeded | Today at 3:28 F
TRTCDemo
config.json
TRTCDemo
Products
Frameworks
libc++.tbd
TXLiteAVSD...C.framework
General
Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
TRTCDemo
TARGETS
TRTCDemo
Access WiFi Information
OFF
App Groups
OFF
Apple Pay
OFF
Associated Domains
OFF
AutoFill Credential Provider
OFF
Background Modes
ON
Modes:
Audio, AirPlay, and Picture in Picture
Location updates
Voice over IP
Newsstand downloads
External accessory communication
Uses Bluetooth LE accessories
Acts as a Bluetooth LE accessory
Background fetch
Remote notifications
Steps:
Add the Required Background Modes key to your info plist file

# 快速集成(Android)

最近更新时间：2022-03-03 14:38:25

本文主要介绍如何快速地将腾讯云 TRTC SDK(Android) 集成到您的项目中，只要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 3.5+。
- Android 4.1（SDK API 16）及以上系统。

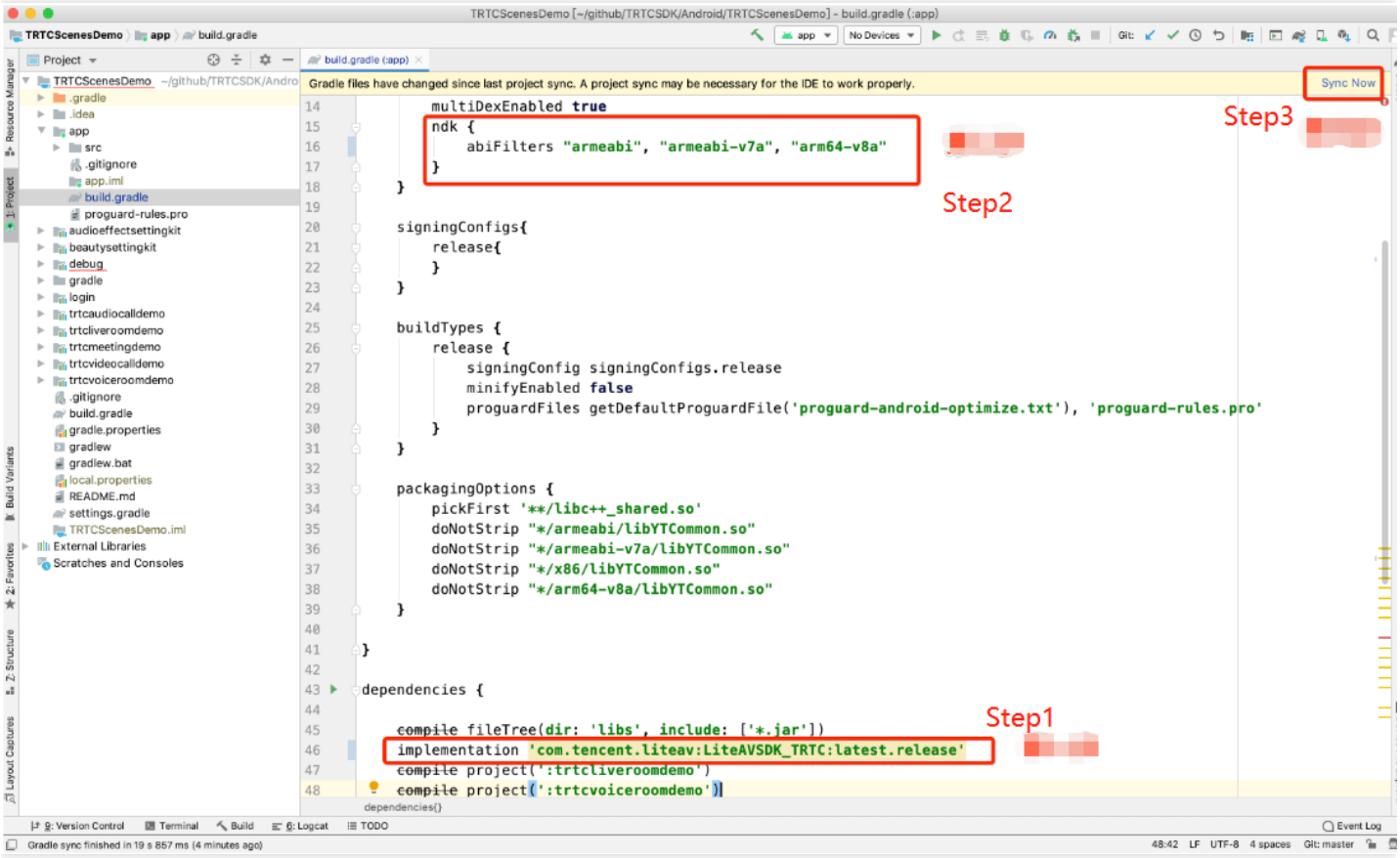
## 集成 SDK（aar）

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）

TRTC SDK 已经发布到 mavenCentral 库，您可以通过配置 gradle 自动下载更新。
只需要用 Android Studio 打开需要集成 SDK 的工程（本文以 TRTCScenesDemo 为例），然后通过简单的三个

步骤修改 app/build.gradle 文件，就可以完成 SDK 集成：




1. 在 dependencies 中添加 TRTCSDK 的依赖。

- 若使用3.x版本的 com.android.tools.build:gradle 工具，请执行以下命令：

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

- 若使用2.x版本的 com.android.tools.build:gradle 工具，请执行以下命令：

```
dependencies {
compile 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'
}
```

2. 在 defaultConfig 中，指定 App 使用的 CPU 架构。

```
defaultConfig {
ndk {
```

```
abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
}
```

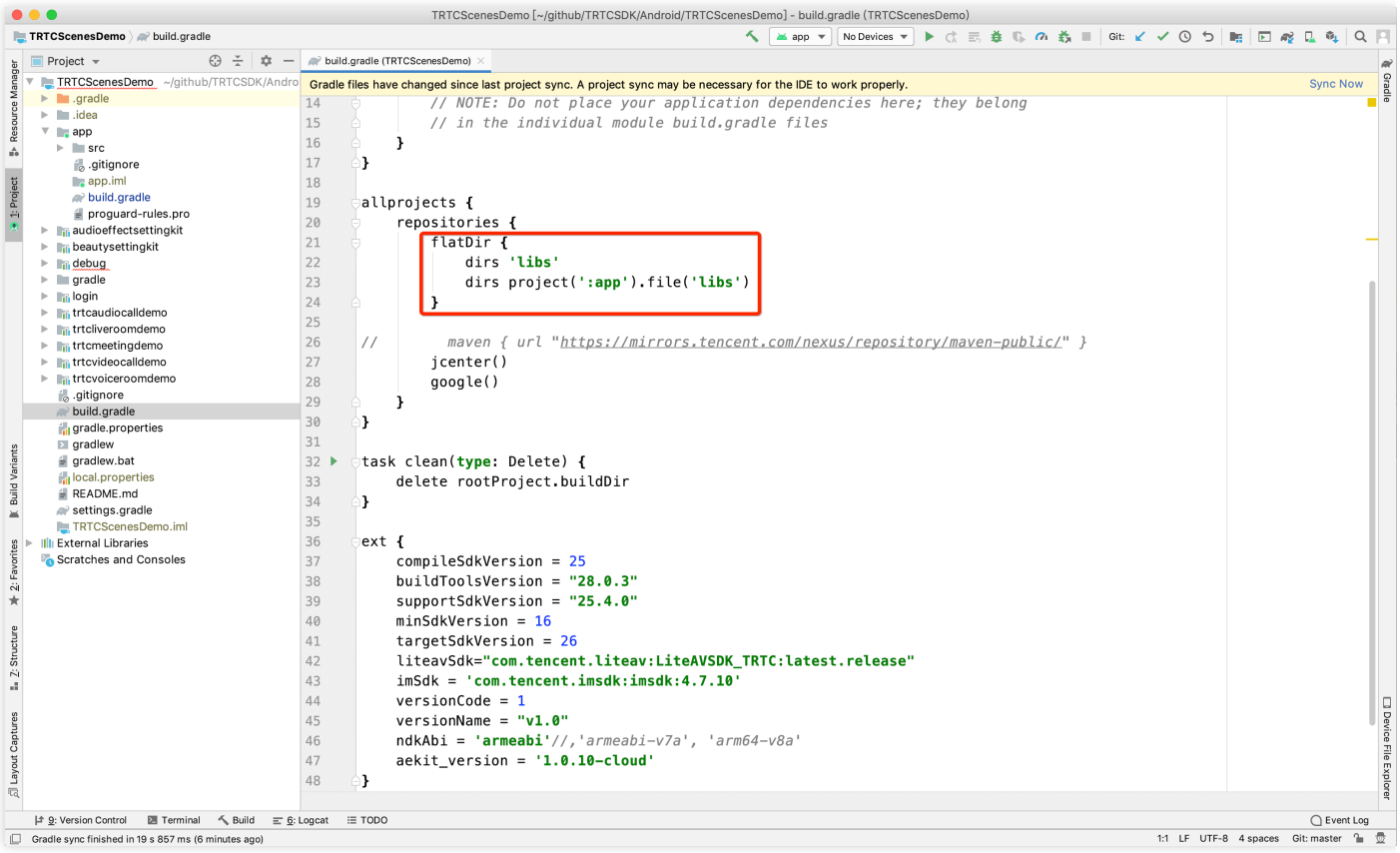
说明：
目前 TRTC SDK 支持 armeabi， armeabi-v7a 和 arm64-v8a。

3. 单击【Sync Now】，自动下载 SDK 并集成到工程里。

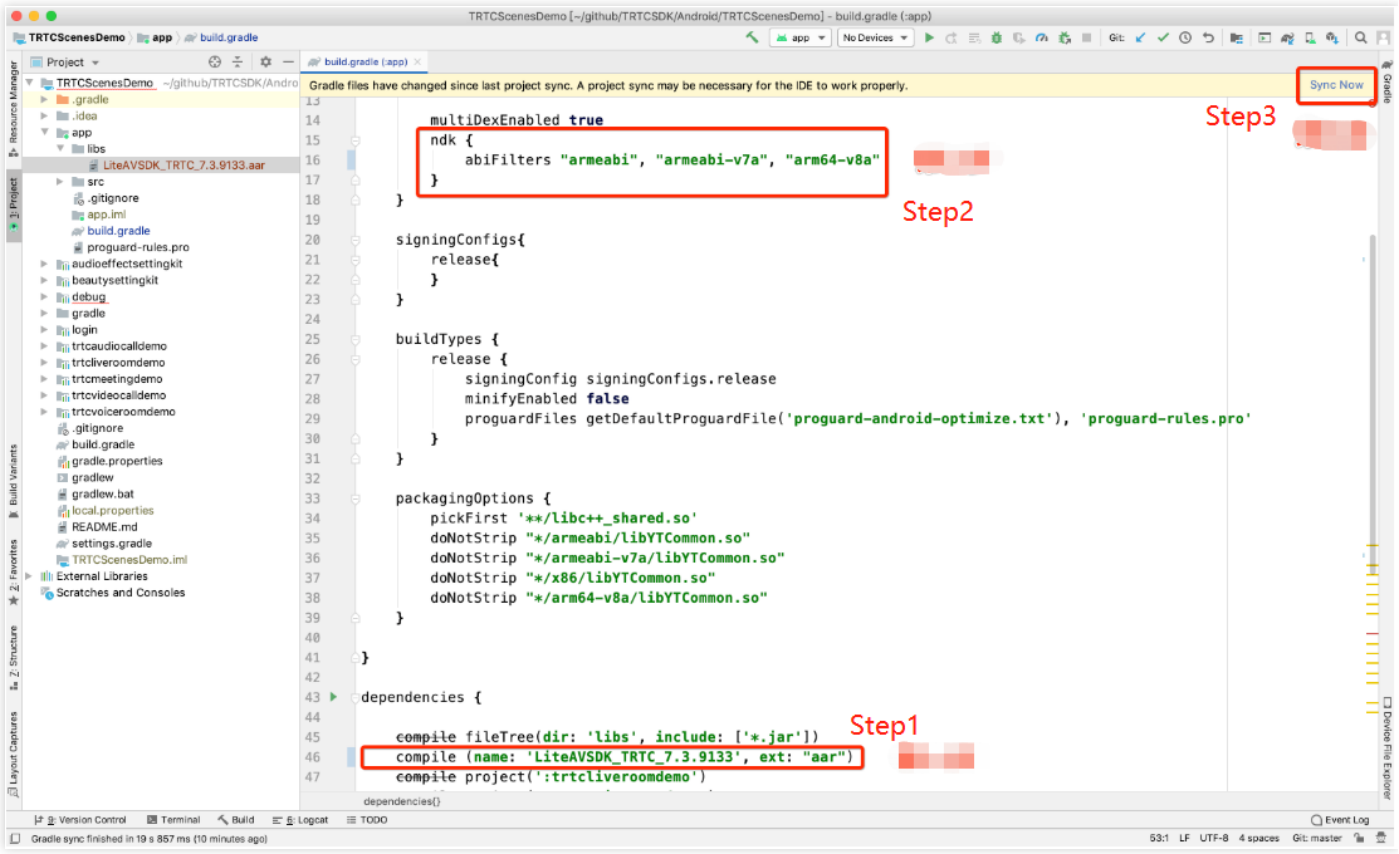
### 方法二：手动下载（aar）

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

1. 下载最新版本 TRTC SDK。
2. 将下载到的 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。



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




5. 在 app/build.gradle的defaultConfig 中，指定 App 使用的 CPU 架构。

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

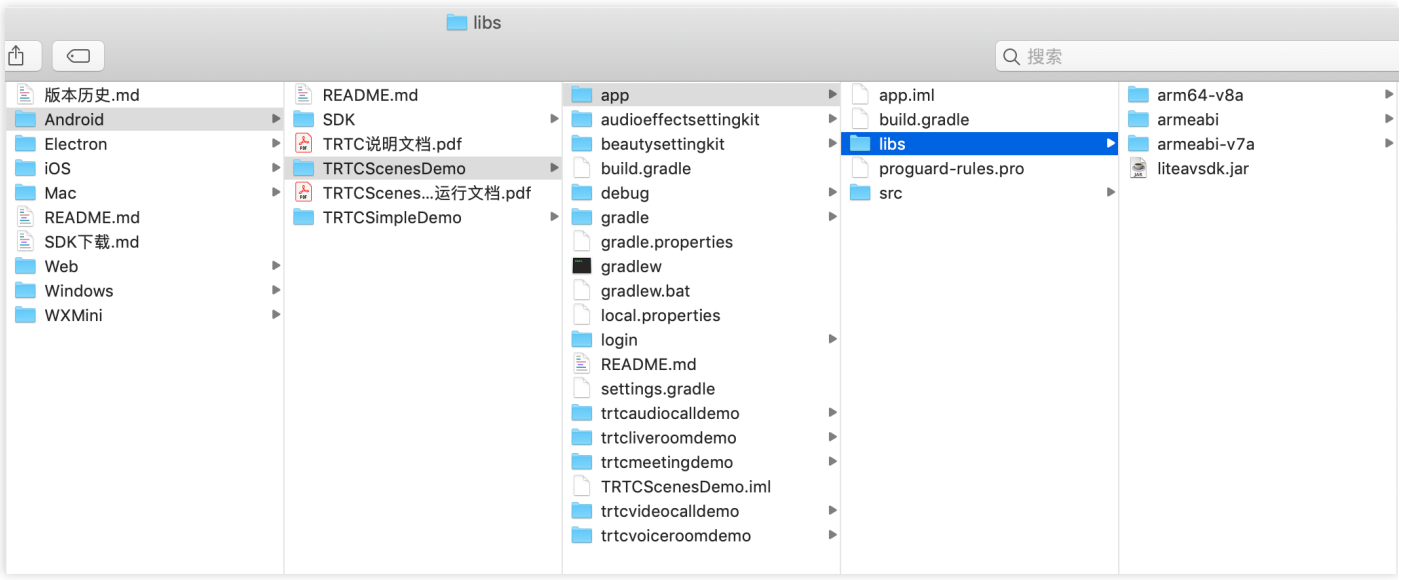
> 说明：
> 目前 TRTC SDK 支持 armeabi， armeabi-v7a 和 arm64-v8a。

6. 单击【Sync Now】，完成 TRTC SDK 的集成工作。

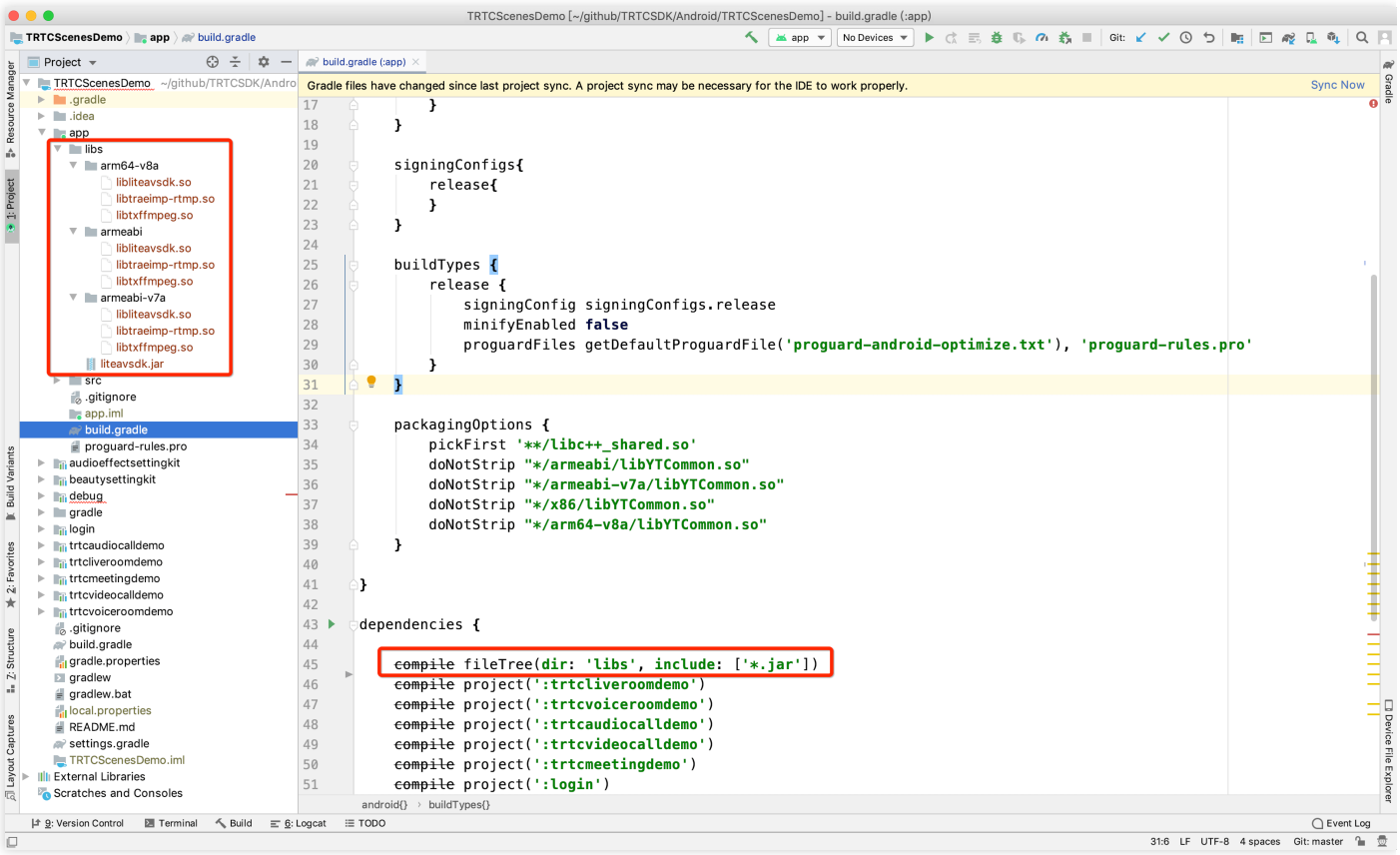
# 集成 SDK（jar）

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 TRTC SDK：

1. 下载 最新版本的 jar 压缩包，文件路径为 `SDK/LiteAVSDK_TRTC_xxx.zip` （其中 xxx 为 TRTC SDK 的版本号）。
2. 解压后得到 libs 目录，里面主要包含 jar 文件和 so 文件夹。
3. 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。



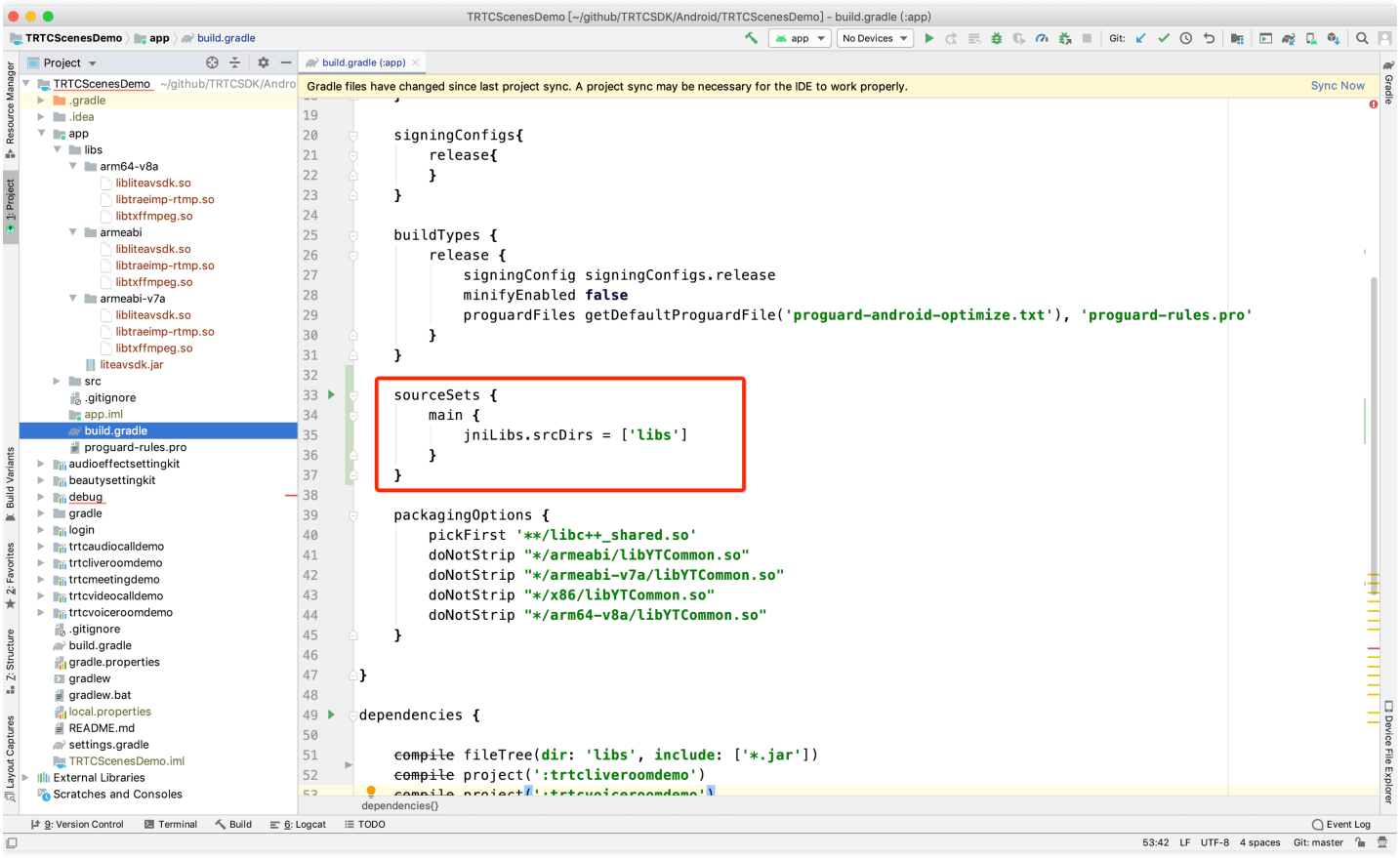

4. 在 app/build.gradle 中，添加引用 jar 库的代码。




```
        }
    }

    signingConfigs{
        release{
        }
    }

    buildTypes {
        release {
            signingConfig signingConfigs.release
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }

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

}

dependencies {

    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile project(':trtcliveroomdemo')
    compile project(':trtcvoiceroomdemo')
    compile project(':trtcaudiocalldemo')
    compile project(':trtcvideocalldemo')
    compile project(':trtcmeetingdemo')
    compile project(':login')
```

5. 在 app/build.gradle 中，添加引用 so 库的代码。

```
sourceSets {
main {
jniLibs.srcDirs = ['libs']
}
}
```




6. 在 app/build.gradle 的 defaultConfig 中，指定 App 使用的 CPU 架构。

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

> 说明：
> 目前 TRTC SDK 支持 armeabi， armeabi-v7a 和 arm64-v8a。

7. 单击【Sync Now】，完成 TRTC SDK 的集成工作。

# 配置 App 权限

在 AndroidManifest.xml 中配置 App 的权限，TRTC SDK 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-feature android:name="android.hardware.camera.autofocus" />
```

注意：
请勿设置 `android:hardwareAccelerated="false"`，关闭硬件加速之后，会导致对方的视频流无法渲染。

# 设置混淆规则

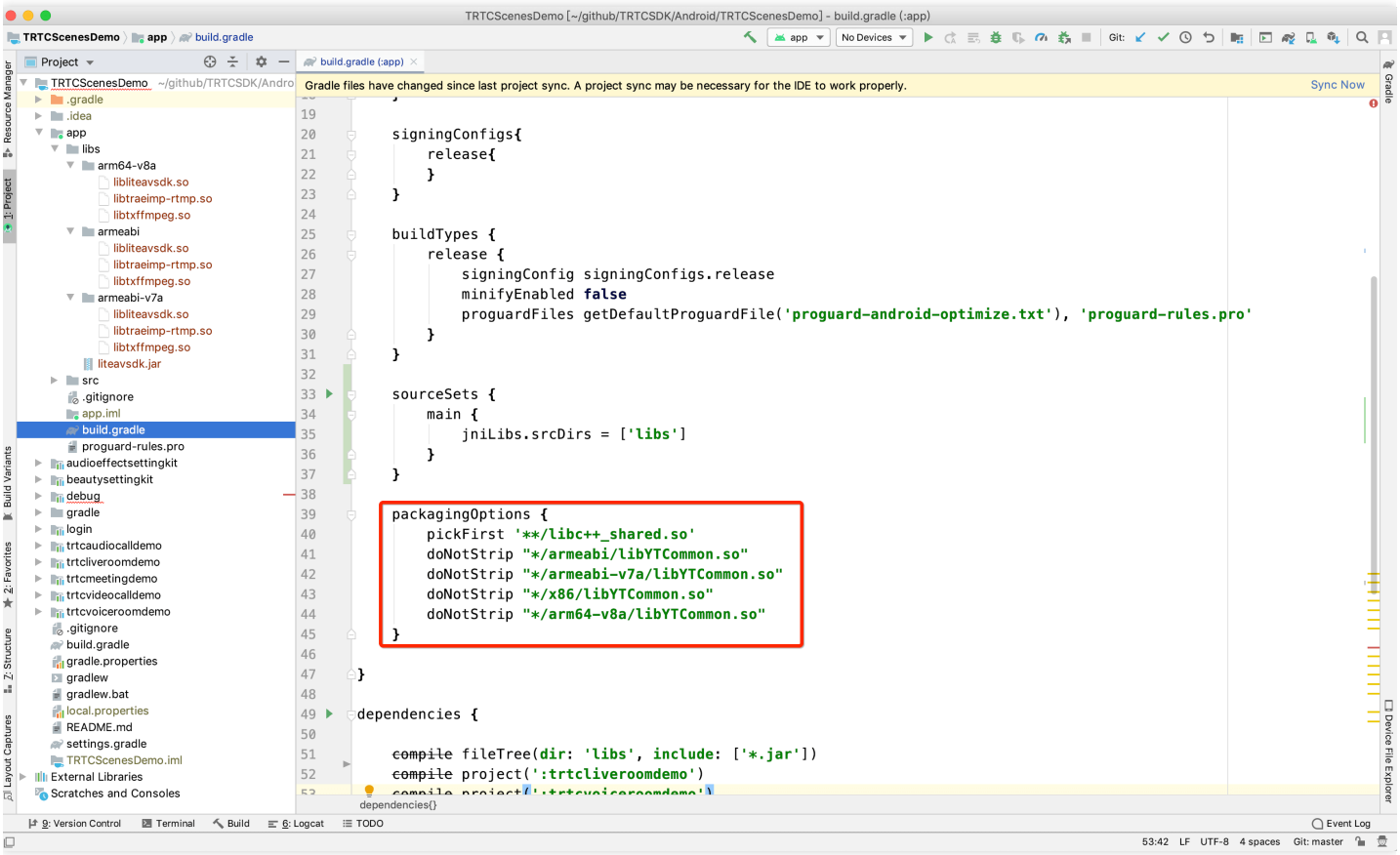
在 proguard-rules.pro 文件，将 TRTC SDK 相关类加入不混淆名单：

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

# 设置 App 打包参数

在 app/build.gradle 下，添加如下信息：

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




# 通过 C++ 接口使用 SDK（可选）

如果您更倾向于使用 C++ 接口，而不是 Java 进行开发，可以执行此步骤；如果您仅使用 Java 语言来调用 TRTC SDK，请忽略此步。

1. 首先需要根据上文的指引，通过导入 jar 和 so 库的方式集成 TRTC SDK。

2. 拷贝头文件：将 SDK 中的 C++ 头文件拷贝到项目中（路径为：`SDK/LiteAVSDK_TRTC_xxx/libs/include`），并在 CMakeLists.txt 中配置 include 文件夹路径及 so 库的动态链接。

```
cmake_minimum_required(VERSION 3.6)
# 配置 C++ 接口头文件路径
include_directories(
${CMAKE_CURRENT_SOURCE_DIR}/include # 拷贝自 SDK/LiteAVSDK_TRTC_xxx/libs/include
)
add_library(
native-lib
SHARED
native-lib.cpp)
```

```
# 配置 libliteavsdk.so 动态库路径
add_library(libliteavsdk SHARED IMPORTED)
set_target_properties(libliteavsdk PROPERTIES IMPORTED_LOCATION ${CMAKE_CURRENT_SOURCE_DIR}
/../../../libs/${ANDROID_ABI}/libliteavsdk.so)
find_library(
log-lib
log)
# 配置 libliteavsdk.so 动态链接
target_link_libraries(
native-lib
libliteavsdk
${log-lib})
```

3. 使用命名空间：C++ 全平台接口的方法、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间

```
using namespace trtc;
```

说明：

- 配置 Android Studio C/C++ 开发环境具体可以参考 Android Studio 官方文档：向 Android 项目添加 C 和 C++ 代码 。
- 目前只有 TRTC 版本的 SDK 支持 C++ 接口；对于 C++ 接口的使用方式，请参见 全平台（C++）API 概览。

# 快速集成(Mac)

最近更新时间：2022-04-02 17:49:38

本文主要介绍如何快速地将腾讯云 TRTC SDK（Mac）集成到您的项目中，只要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+。
- OS X10.10+ 的 Mac 真机。
- 项目已配置有效的开发者签名。

## 集成 TRTC SDK

您可以选择使用 CocoaPods 自动加载的方式，或者手动先下载 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 LiteAV SDK 的 pod 路径。

```
platform :osx, '10.10'

target 'Your Target' do
pod 'TXLiteAVSDK_TRTC_Mac', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsdkspec/TXLit
```

```
eAVSDK_TRTC_Mac.podspec'
end
```

- 方式二：使用 CocoaPod 官方源，支持选择版本号。

```
platform :osx, '10.10'
source 'https://github.com/CocoaPods/Specs.git'

target 'Your Target' do
pod 'TXLiteAVSDK_TRTC_Mac'
end
```

#### 4. 安装与更新 SDK

在终端窗口中输入如下命令执行安装 TRTC SDK：

```
pod install
```

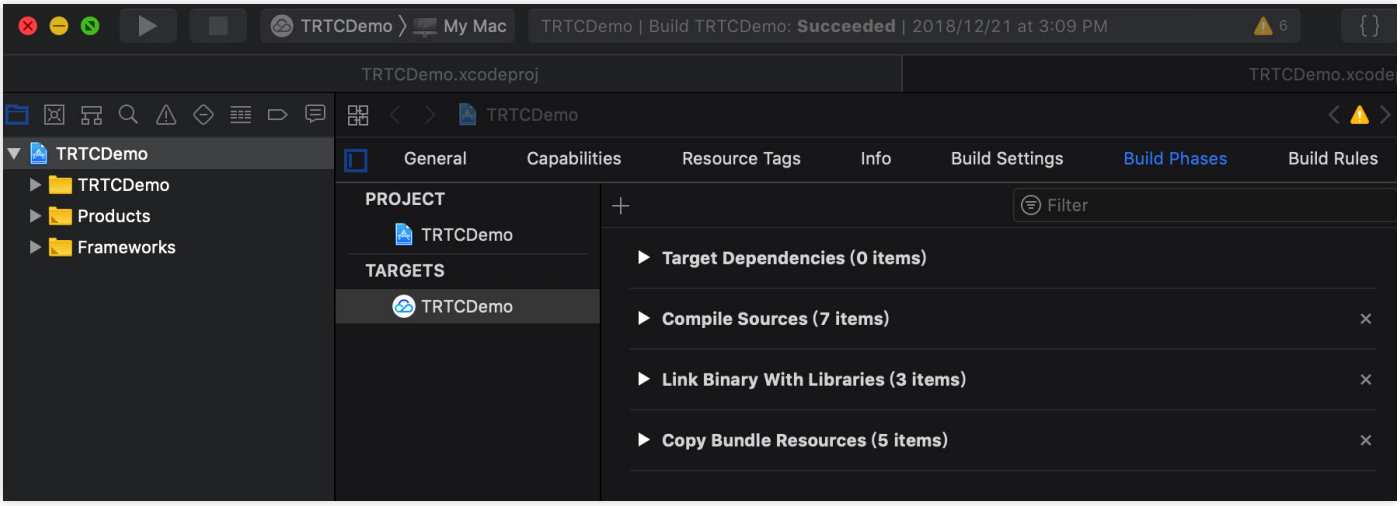
或使用以下命令更新本地库版本：

```
pod update
```

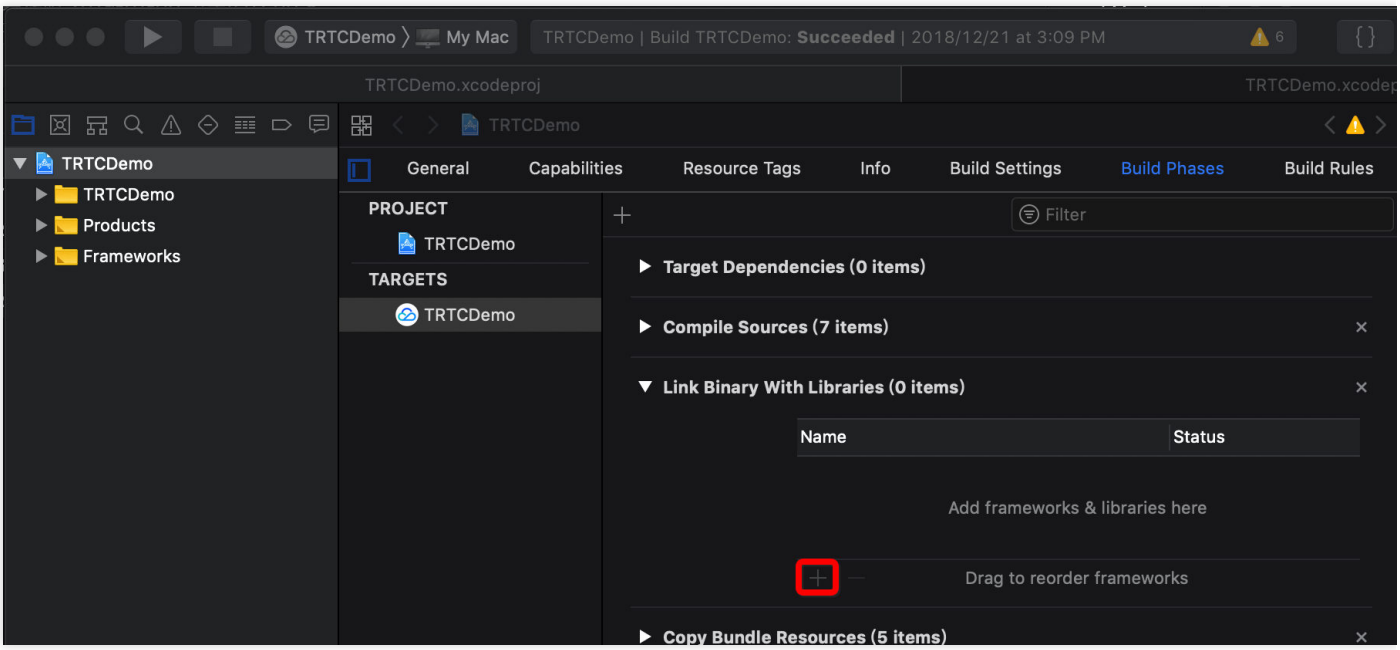
pod 命令执行完后，会生成集成了 SDK 的 `.xcworkspace` 后缀的工程文件，双击打开即可。

### 手动集成

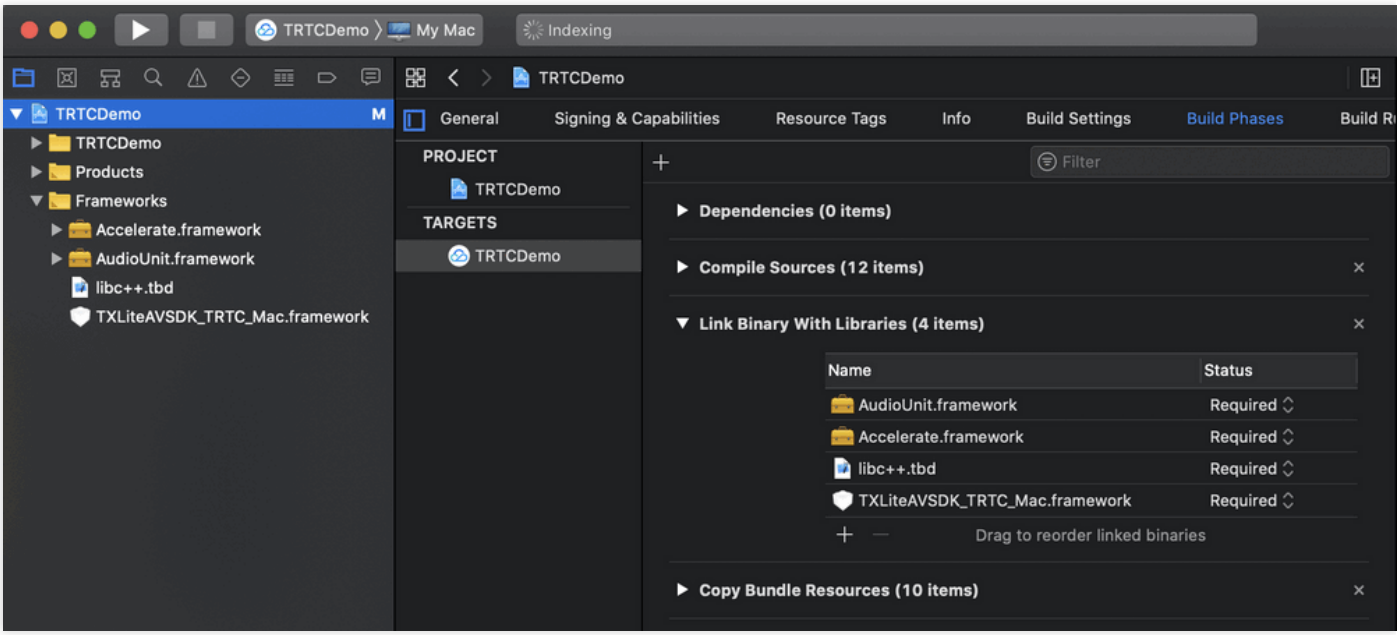
1. 下载 TRTC-SDK 的 Mac 版本。
2. 打开您的 Xcode 工程项目，将第一步中下载的 framework 导入到您的工程。
3. 选择要运行的 target，选中 Build Phases 项。



4. 单击 **Link Binary with Libraries** 项展开，单击底下的+号图标去添加依赖库。



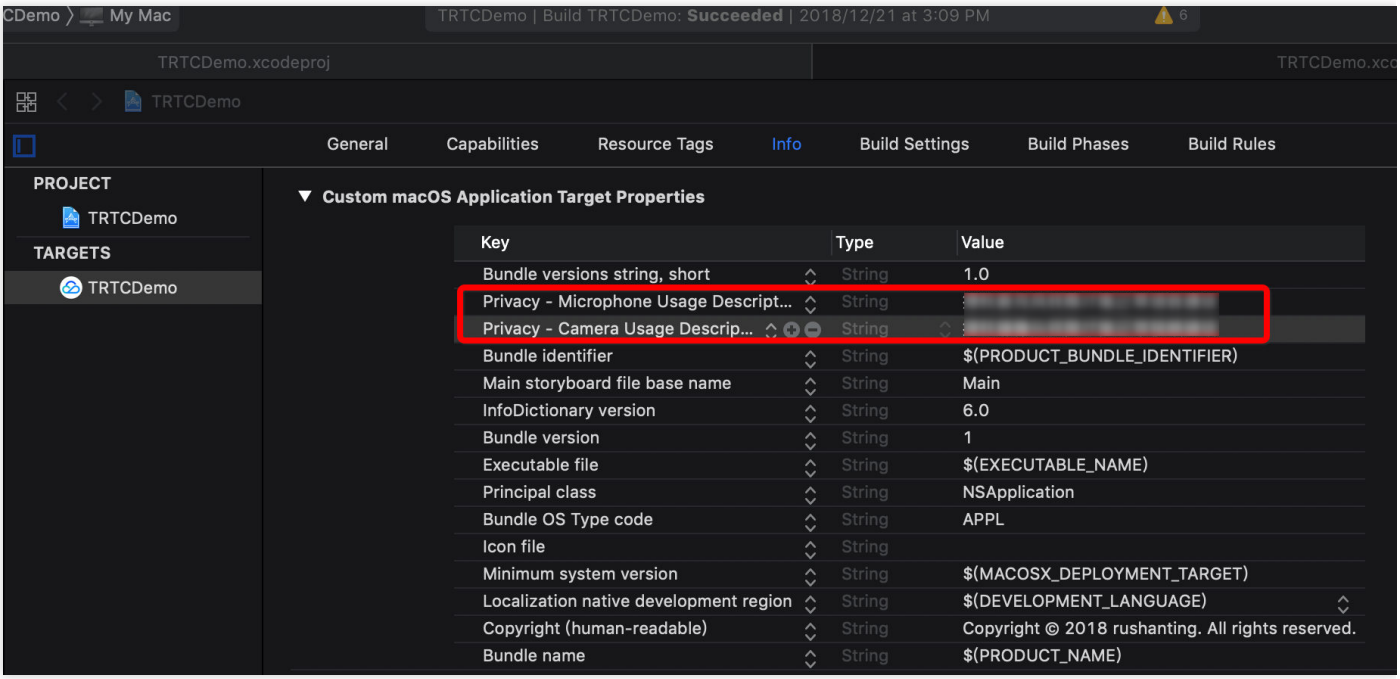

5. 依次添加所下载的 SDK Framework 及其所需依赖库：`AudioUnit.framework`、`libc++.tbd` 和 `Accelerate.framework`。
   添加后如下图所示：




# 授权摄像头和麦克风使用权限

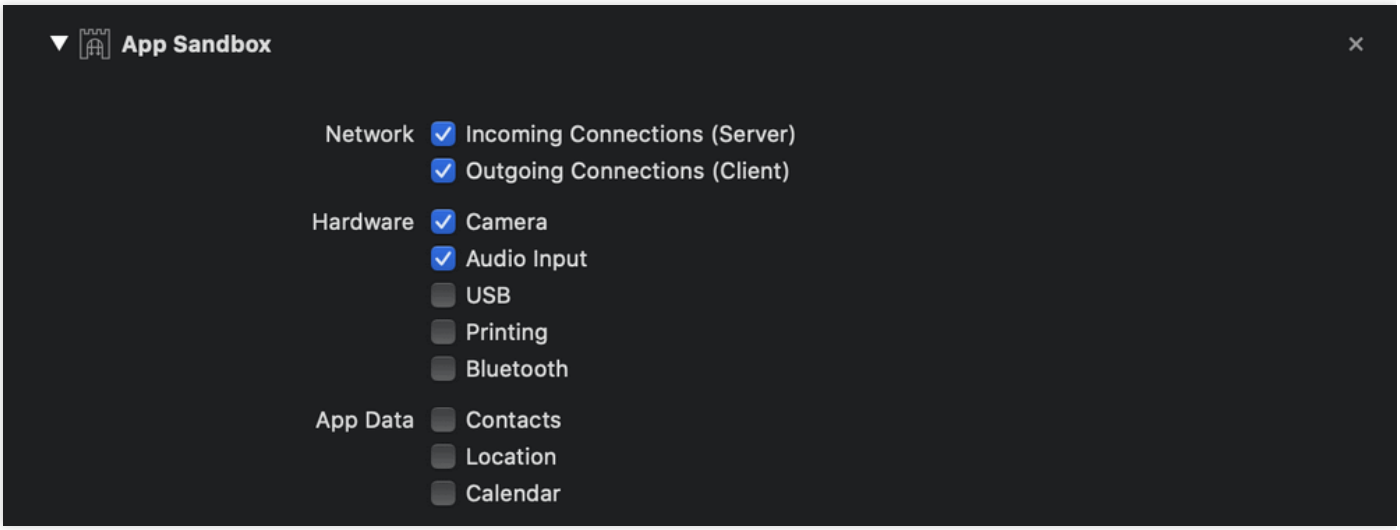
使用 SDK 的音视频功能，需要授权麦克风和摄像头的使用权限。在 App 的 Info.plist 中添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

- **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 。
  如下图所示：




如果 App 启用了 **App Sandbox** 或 **Hardened Runtime**，需要勾选上 `Network`、`Camera` 和 `Audio Input` 这几个选项。

- App Sandbox 配置如下图所示：



- Hardened Runtime 配置如下图所示：

# 引用 TRTC SDK

TRTC SDK 支持两种调用方式，您可以任选一种：

### 方案一：通过 Objective-C 或 Swift 接口引用 TRTC SDK

在 Objective-C 或 Swift 代码中使用 SDK 有两种方式：

- **模块引用**：在项目需要使用 SDK API 的文件里，添加模块引用。

```
@import TXLiteAVSDK_TRTC_Mac;
```

- **头文件引用**：在项目需要使用 SDK API 的文件里，引入具体的头文件。

```
#import TXLiteAVSDK_TRTC_Mac/TRTCCloud.h
```

### 方案二：通过 C++ 接口引用 TRTC SDK

1. **引用头文件**：如果您要使用 C++ 接口来开发Mac应用，请引用 `TXLiteAVSDK_TRTC_Mac.framework/Headers/cpp_interface` 目录下的头文件。

```
#include TXLiteAVSDK_TRTC_Mac/cpp_interface/ITRTCCloud.h
```

2. **使用命名空间**：C++ 全平台接口的接口、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间。

```
using namespace trtc;
```

> 说明：
> 对于 C++ 接口的使用方式，请参见 全平台（C++）API 概览。

# 快速集成(Windows)

最近更新时间：2021-04-12 15:39:48

本文介绍如何快速地将腾讯云 TRTC SDK（Windows C# 和 C++ 版本）集成到项目中。

## 开发环境要求

- 操作系统：Windows 7及以上版本。
- 开发环境：Visual Studio 2010及以上版本，推荐使用 Visual Studio 2015。
- 开发框架：.Net Framework 4.0及以上版本。

## 集成 TRTC C# SDK

本节以创建一个简单的 Winform 项目为例，介绍如何在 Visual Studio 工程中集成 C# SDK。

### 步骤1：下载 Windows SDK

下载 SDK，解压并打开文件，包含以下部分：

| 目录名 | 说明 |
| --- | --- |
| xxxDemo | C++ Demo 源码和 C# Demo 源码 |
| CPlusPlus | C++版32位/64位依赖的 SDK 库文件 |
| CSharp | C#版32位/64位依赖的 SDK 库文件 |

本文示例中，您只需要引用 SDK 目录下 C# 版的 SDK 文件即可。

### 步骤2：新建工程

打开 Visual Studio，新建名为 `TRTCCSharpDemo` 的 Winform 应用程序。

### 步骤3：拷贝文件

将解压后的 SDK 文件夹拷贝至 `TRTCCSharpDemo.csproj` 所在目录。

**说明：**

当只需要 C# SDK时，可以将 SDK 路径下的 CPlusPlus 目录删除。

### 步骤4：修改工程配置

#### 步骤4.1：添加引用

1. 在 Visual Studio 的【生成】目录下找到【配置管理器】并打开。
2. 在【活动解决方案平台】下拉框中选择【新建】，弹出【新建解决方案平台】对话框。
3. 输入或选择新平台，单击【确定】。
4. 根据实际需求重复 步骤2 - 步骤3 新建需要支持的解决方案平台。
5. 打开 TRTCCSharpDemo 项目所在的文件夹，并用文本编辑器编辑 `TRTCCSharpDemo.csproj` 文件。
6. 在 `TRTCCSharpDemo.csproj` 文件中的标签 `<itemGroup>` 下添加以下内容：

```
//添加对不同平台下的引用
<Reference Include="ManageLiteAV" Condition="'$(Platform)' == 'x64'">
<HintPath>SDK¥CSharp¥Win64¥lib¥ManageLiteAV.dll</HintPath>
</Reference>
<Reference Include="ManageLiteAV" Condition="'$(Platform)' == 'AnyCPU'">
<HintPath>SDK¥CSharp¥Win64¥lib¥ManageLiteAV.dll</HintPath>
</Reference>
<Reference Include="ManageLiteAV" Condition="'$(Platform)' == 'x86'">
<HintPath>SDK¥CSharp¥Win32¥lib¥ManageLiteAV.dll</HintPath>
</Reference>
```

#### 步骤4.2：添加 copy 命令

1. 打开 TRTCCSharpDemo 属性页，选择【解决方案资源管理器】>【TRTCCSharpDemo 工程的右键菜单】>【属性】。
2. 在【生成事件】>【后期生成事件命令行】中添加以下命令，实现在编译完成后自动将不同平台下的 SDK 的 .dll 文件拷贝到程序的运行目录下，如下图所示：

```
set Platform=Win64
SETLOCAL ENABLEDELAYEDEXPANSION
if $(PlatformName)==x86 (
set Platform=Win32
)
copy /Y "$(ProjectDir)SDK¥CSharp¥!Platform!¥lib¥*.dll" "$(ProjectDir)$(OutDir)"
ENDLOCAL
```

**步骤4.3：修改调试环境**

打开 TRTCDemo 属性页，选择【生成】，将【平台(M)】与顶部菜单栏中的解决方案平台设置为一致，如下图所示：

### 步骤5：打印 SDK 版本号

1. 在 Form1.cs 的设计器中添加一个 label 控件，如下图所示：

2. 打开 Form1.cs 代码文件，添加以下代码：

```
using System.Windows.Forms;
using ManageLiteAV; // 1.添加命名空间引用

namespace TRTCCSharpDemo
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
// 2.获取 ITRTCCloud 实例，打印 SDK 版本号
ITRTCCloud lTRTCCloud = ITRTCCloud.getTRTCShareInstance();
this.label1.Text = "SDK version : " + lTRTCCloud.getSDKVersion();
// 3.结束使用时需手动摧毁 ITRTCCloud 实例
ITRTCCloud.destroyTRTCShareInstance();
}
}
}
```

3. 按 F5 运行，打印 SDK 的版本号，如下图所示：


# 集成 TRTC C++ SDK

本节通过创建一个简单的 MFC 项目，介绍如何在 Visual Studio 工程中集成 C++ SDK。

### 步骤1：下载 SDK

下载 SDK，解压并打开，包含以下几个部分：

| 目录名 | 说明 |
| --- | --- |
| include | 带有详细接口注释的 API 头文件 |
| lib | 编译用的 .lib 文件和运行时加载的 .dll 文件 |

### 步骤2：新建工程

打开 Visual Studio，新建一个名字叫 TRTCDemo 的 MFC 应用程序，如下图所示：

为了便于介绍如何快速集成，在向导的**应用程序类型**页面，我们选择比较简单的**基于对话框**类型，如下图所示：

其他的向导配置，请选择默认的配置即可。

### 步骤3：拷贝文件

将解压后的 LiteAVSDK 文件夹拷贝到 TRTCDemo.vcxproj 所在目录下，如下图所示：

### 步骤4：修改工程配置

打开 TRTCDemo 属性页，在【解决方案资源管理器】>【TRTCDemo 工程的右键菜单】>【属性】，请按照以下步骤进行配置：

1. **添加包含目录：**
   在【C/C++】>【常规】>【附件包含目录】，添加 SDK 头文件目录 `$(ProjectDir)LiteAVSDK¥include` 和 `$(ProjectDir)LiteAVSDK¥include¥TRTC`，如下图所示：

2. **添加库目录：**
   在【链接器】>【常规】>【附加库目录】，添加 SDK 库目录 `$(ProjectDir)LiteAVSDK¥lib`，如下图所示：

3. **添加库文件：**
   在【链接器】>【输入】>【附加依赖项】，添加 SDK 库文件 `liteav.lib`，如下图所示：

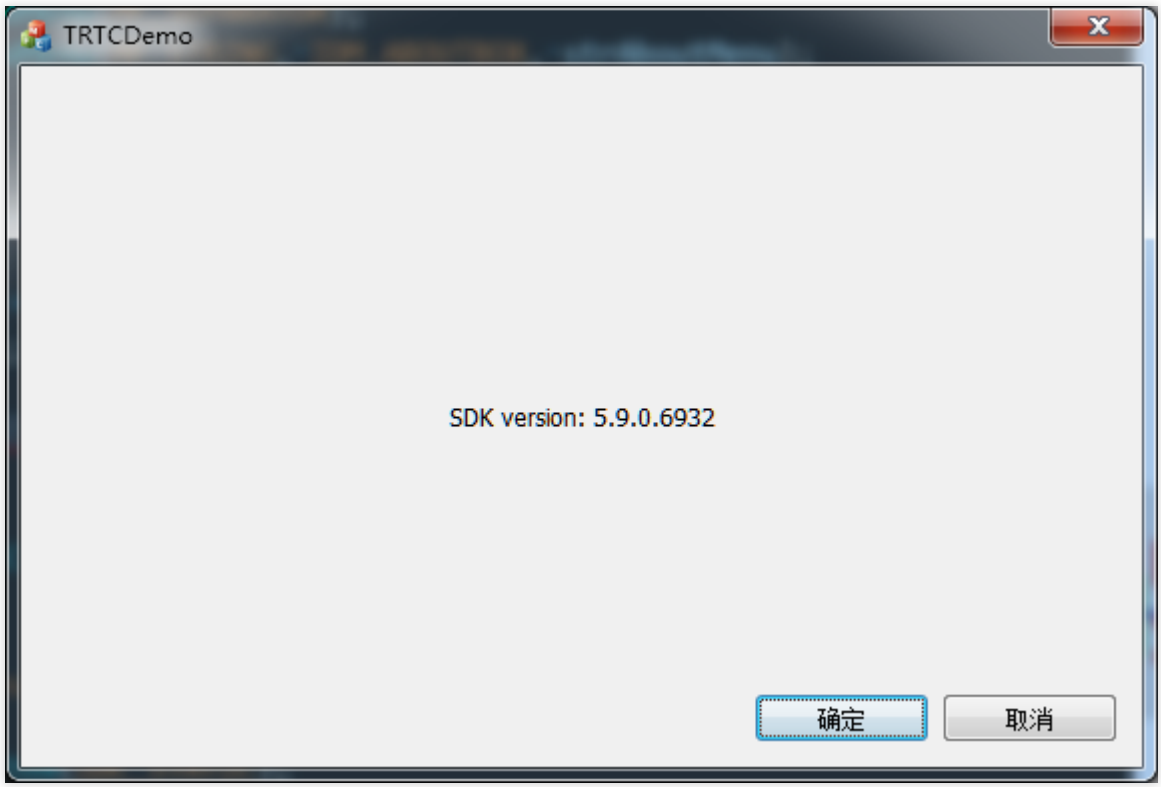
4. **添加 copy 命令：**
   在【生成事件】>【后期生成事件】>【命令行】，添加拷贝命令 `copy /Y "$(ProjectDir)LiteAVSDK¥lib¥¥¥*.dll" "¥$(OutDir)"`，能够在编译完成后，自动将 SDK 的 .dll 文件拷贝到程序的运行目录下，如下图所示：

### 步骤5：打印 SDK 版本号

- 在 `CTRTCDemoDlg::OnInitDialog` 函数中，添加下面的测试代码：
  - c++

```
CWnd *pStatic = GetDlgItem(IDC_STATIC);
pStatic->SetWindowTextW(szText);
:::
```

- 按键盘 F5 运行，打印 SDK 的版本号，如下图所示：




# 常见问题

- 若出现以下错误，请按照 修改工程配置，检查 SDK 引用是否添加到工程中。

```
错误 CS0246 未能找到类型或命名空间名“ManageLiteAV”(是否缺少 using 指令或程序集引用?)
```

- 若出现以下错误，请按照 修改工程配置，检查是否修改工程运行平台环境为解决方案当前目标平台 。

```
System.BadImageFormatException: “未能加载文件或程序集“ManageLiteAV, Version=2.0.7152.18518, Culture=neutral, PublicKeyToken=null”或它的某一个依赖项。试图加载格式不正确的程序。”
```

- 若出现以下错误，请按照 修改工程配置，检查是否正确添加生成事件到运行目录中。

```
System.IO.FileNotFoundException: “未能加载文件或程序集“ManageLiteAV.dll”或它的某一个依赖项。找不到指定的模块。”
```

- 由于 Windows 不同版本可能存在兼容性问题，目前在 C# SDK 中新增了解决兼容性问题的 dll 文件，文件清单如下图所示。

- 若出现以下错误，请按照前面的工程配置，检查 SDK 头文件的目录是否正确添加。

```
fatal error C1083: 无法打开包括文件: “TRTCCloud.h”: No such file or directory
```

- 若出现以下错误，请按照前面的工程配置，检查 SDK 库目录和库文件是否正确添加。

```
error LNK2019: 无法解析的外部符号 "__declspec(dllimport) public: static class TXString __cdecl TRTCCloud::getSDKVersion(void)" (__imp_?getSDKVersion@TRTCCloud@@SA?AVTXString@@XZ)，该符号在函数 "protected: virtual int __thiscall CTRTCDemoDlg::OnInitDialog(void)" (?OnInitDialog@CTRTCDemoDlg@@MAEHXZ) 中被引用
```

# 快速集成(Web)

最近更新时间：2022-02-14 11:38:24

本文主要介绍如何快速地将腾讯云 TRTC Web SDK 集成到您的项目中。

## 支持的平台

WebRTC 技术由 Google 最先提出， Chrome 、Edge 、 Firefox、Safari 、Opera浏览器等均已支持，腾讯云 TRTC Web SDK基于WebRTC封装，腾讯云 TRTC Web SDK 详细支持度表格请参见 支持的平台。

- 如果您的应用场景主要为教育场景，那么教师端推荐使用稳定性更好的 Electron 解决方案，支持大小双路画面，更灵活的屏幕分享方案以及更强大的弱网络恢复能力。

注意：

- 您可以在浏览器中打开 TRTC Web SDK 能力测试页面 检测当前浏览器是否支持 WebRTC 所有能力。例如 WebView 等浏览器环境。
- 由于 H.264 版权限制，华为 Chrome 88 以下版本，无法使用 H264 编码（即无法推流）。如果您希望在华为设备 Chrome 浏览器中，使用 TRTC Web SDK 推流，请提交工单申请开通 VP8 编解码。

## URL 域名协议限制

| 应用场景 | 协议 | 接收（播放） | 发送（上麦） | 屏幕分享 | 备注 |
| --- | --- | --- | --- | --- | --- |
| 生产环境 | HTTPS 协议 | 支持 | 支持 | 支持 | 推荐 |
| 生产环境 | HTTP 协议 | 支持 | 不支持 | 不支持 | |
| 本地开发环境 | http://localhost | 支持 | 支持 | 支持 | 推荐 |
| 本地开发环境 | http://127.0.0.1 | 支持 | 支持 | 支持 | |
| 本地开发环境 | http://[本机IP] | 支持 | 不支持 | 不支持 | |
| 本地开发环境 | file:/// | 支持 | 支持 | 支持 | |

## 防火墙限制

TRTC Web SDK 依赖以下端口进行数据传输，请将其加入防火墙白名单。

- TCP 端口：8687
- UDP 端口：8000，8080，8800，843，443，16285
- 域名：qcloud.rtc.qq.com

# 集成 TRTC Web SDK

### NPM 集成

1. 您需要在项目中使用 npm 安装 SDK 包。

```
npm install trtc-js-sdk --save
```

2. 在项目脚本里引入模块。

```
import TRTC from 'trtc-js-sdk';
```

### Script 集成

您只需要在您的 Web 页面中添加如下代码即可：

```
<script src="trtc.js"></script>
```

# 相关资源

SDK 下载地址：单击下载 。

更详细的初始化流程和 API 使用介绍请参见以下指引：

| 功能 | Sample Code 指引 |
| --- | --- |
| 基础音视频通话 | 指引链接 |
| 互动直播 | 指引链接 |
| 切换摄像头和麦克风 | 指引链接 |
| 设置本地视频属性 | 指引链接 |
| 动态关闭打开本地音频或视频 | 指引链接 |

| 功能 | Sample Code 指引 |
| --- | --- |
| 屏幕分享 | 指引链接 |
| 音量大小检测 | 指引链接 |
| 自定义采集与自定义播放渲染 | 指引链接 |
| 房间内上行用户个数限制 | 指引链接 |
| 背景音乐和音效实现方案 | 指引链接 |
| 通话前环境与设备检测 | 指引链接 |
| 通话前的网络质量检测 | 指引链接 |
| 检测设备插拔行为 | 指引链接 |
| 实现推流到 CDN | 指引链接 |
| 开启大小流传输 | 指引链接 |
| 开启美颜 | 指引链接 |
| 开启水印 | 指引链接 |
| 实现跨房连麦 | 指引链接 |

说明：
单击查看 更多能力。

# 快速集成(Electron)

最近更新时间：2022-01-19 15:23:31

本文主要介绍如何快速地将腾讯云 TRTC Electron SDK 集成到您的项目中。

## 支持的平台

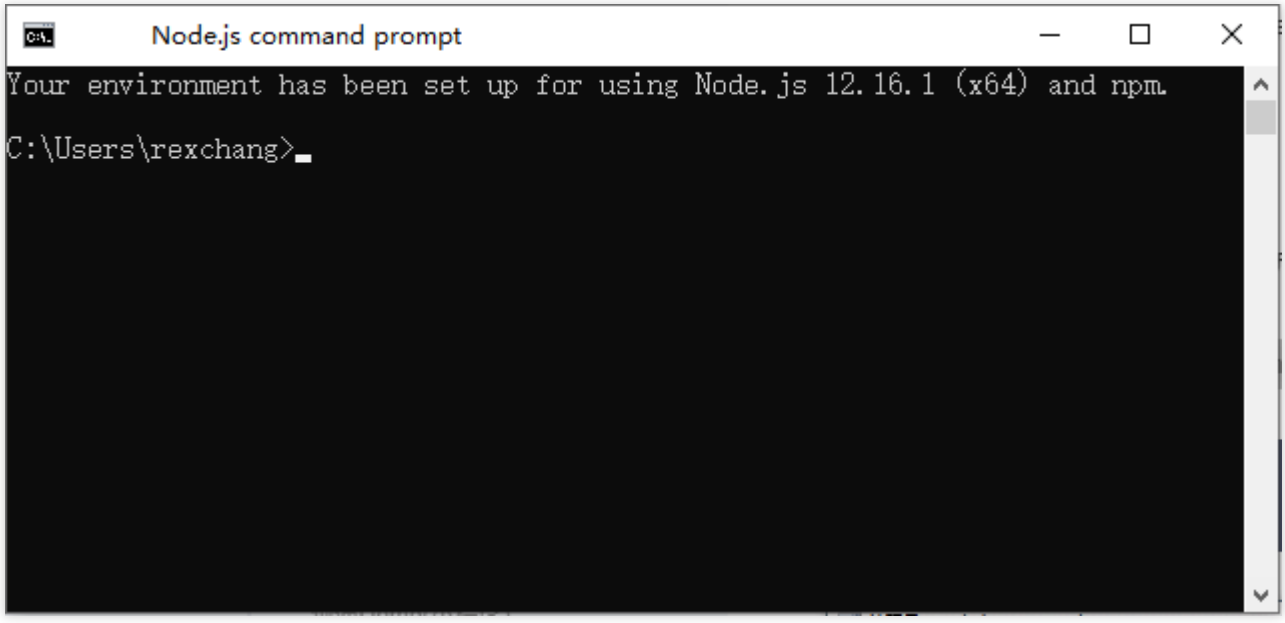
- Windows（PC）
- Mac

## 集成 TRTC Electron SDK

### 步骤1：安装 Node.js

- Windows平台安装指引
- MacOS平台安装指引

1. 根据 Windows 操作系统选择下载最新版本的 Node.js 安装包 `Windows Installer (.msi) 64-bit`。
2. 打开应用程序列表中的 Node.js command prompt，启动命令行窗口，用于输入后续步骤中的各项命令。




### 步骤2：安装 Electron

在命令行窗口中执行如下命令，安装 Electron，建议版本号 >= 4.0.0。

```
$ npm install electron@latest --save-dev
```

### 步骤3：安装 Electron 版的 TRTC SDK

1. 在您的 Electron 项目中使用 npm 命令安装 SDK 包：

```
$ npm install trtc-electron-sdk@latest --save
```

> 说明：
> TRTC Electron SDK 最新版可在 trtc-electron-sdk 中查看。

2. 在项目脚本里引入模块并使用：

```
const TRTCCloud = require('trtc-electron-sdk').default;
// import TRTCCloud from 'trtc-electron-sdk';
this.rtcCloud = new TRTCCloud();
// 获取 SDK 版本号
this.rtcCloud.getSDKVersion();
```

从v7.9.348起，TRTC Electron SDK 增加了 trtc.d.ts 文件，方便使用 TypeScript 的开发者：

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud: TRTCCloud = new TRTCCloud();
// 获取 SDK 版本号
rtcCloud.getSDKVersion();
```

## 打包可执行程序

### 步骤1：安装打包工具

1. 推荐使用打包工具 `electron-builder` 进行打包，您可以执行如下命令安装 `electron-builder` ：

```
$ npm install electron-builder@latest --save-dev
```

2. 为了正确打包 Electron 版本的 TRTC SDK（也就是 `trtc_electron_sdk.node` 文件），您还需要执行如下命令以安装 `native-ext-loader` 工具：

```
$ npm install native-ext-loader@latest --save-dev
```

### 步骤2：修改 webpack.config.js 配置

`webpack.config.js` 包含了项目构建的配置信息，`webpack.config.js` 文件的位置如下：

- 通常情况下，`webpack.config.js` 位于项目的根目录。
- 使用 `create-react-app` 创建项目的情况下，此配置文件为 `node_modules/react-scripts/config/webpack.config.js`。
- 使用 `vue-cli` 创建项目的情况下，webpack 的配置存放在 `vue.config.js` 配置中的 `configureWebpack` 属性中。
- 如您的工程文件经过了定制化，还请自行查找 webpack 配置。

1. 首先使 `webpack.config.js` 在构建时可以接收名为 `--target_platform` 的命令行参数，以使代码构建过程按不同的目标平台特点正确打包，在 `module.exports` 之前添加以下代码：

```
const os = require('os');
const targetPlatform = (function(){
let target = os.platform();
for (let i=0; i<process.argv.length; i++) {
if (process.argv[i].includes('--target_platform=')) {
target = process.argv[i].replace('--target_platform=', '');
break;
}
}
if (!['win32', 'darwin'].includes) target = os.platform();
return target;
})();
```

> 注意：
> `os.platform()` 返回的结果中，"darwin" 表示 Mac 平台。"win32" 表示 Windows 平台，不论 64 位还是 32 位。

2. 然后在 `rules` 选项中添加以下配置，`targetPlatform` 变量可以使 `rewritePath` 可以根据不同的目标平台切换不同的配置：

```
rules: [
{
test: /¥.node$/,
loader: 'native-ext-loader',
```

```
options: {
rewritePath: targetPlatform === 'win32' ? './resources' : '../Resources'
}
},
]
```

该配置的含义是：

- 打包 Windows 下的 `.exe` 文件时，让 `native-ext-loader` 到 `[应用程序根目录]/resources` 目录下加载 TRTC SDK。
- 打包 Mac 下的 `.dmg` 时，让 `native-ext-loader` 到 `[应用程序目录]/Contents/Frameworsk/../Resources` 目录下加载 TRTC SDK。

还需要在 `package.json` 中的构建脚本中添加 `--target_platform` 参数，将在下一步进行。

### 步骤3：修改 package.json 配置

`package.json` 位于项目的根目录，其中包含了项目打包所必须的信息。但默认情况下， `package.json` 中的路径是需要修改才能顺利实现打包的，我们可以按如下步骤修改此文件：

1. 修改 `main` 配置。

```
// 多数情况下，main 文件名称可以任意配置，例如 TRTCSimpleDemo 中的可以配置为：
"main": "main.electron.js",

// 但是，使用 create-react-app 脚手架创建的项目，main 文件必须配置为：
"main": "public/electron.js",
```

2. 复制以下 `build` 配置，添加到您的 `package.json` 文件中，这是 `electron-builder` 需要读取到的配置信息。

```
"build": {
"appId": "[appId 请自行定义]",
"directories": {
"output": "./bin"
},
"win": {
"extraFiles": [
{
"from": "node_modules/trtc-electron-sdk/build/Release/",
"to": "./resources",
"filter": ["**/*"]
}
]
},
"mac": {
```

```
"extraFiles": [
{
"from": "node_modules/trtc-electron-sdk/build/Release/trtc_electron_sdk.node",
"to": "./Resources"
}
]
}
},
```

3. 在 `scripts` 节点下添加以下构建和打包的命令脚本：

本文以 `create-react-app` 和 `vue-cli` 项目为例，其它工具创建的项目也可以参考此配置：

```
// create-react-app 项目请使用此配置
"scripts": {
"build:mac": "react-scripts build --target_platform=darwin",
"build:win": "react-scripts build --target_platform=win32",
"compile:mac": "node_modules/.bin/electron-builder --mac",
"compile:win64": "node_modules/.bin/electron-builder --win --x64",
"pack:mac": "npm run build:mac &amp;&amp; npm run compile:mac",
"pack:win64": "npm run build:win &amp;&amp; npm run compile:win64"
}

// vue-cli 项目请使用此配置
"scripts": {
"build:mac": "vue-cli-service build --target_platform=darwin",
"build:win": "vue-cli-service build --target_platform=win32",
"compile:mac": "node_modules/.bin/electron-builder --mac",
"compile:win64": "node_modules/.bin/electron-builder --win --x64",
"pack:mac": "npm run build:mac &amp;&amp; npm run compile:mac",
"pack:win64": "npm run build:win &amp;&amp; npm run compile:win64"
}
```

| 参数 | 说明 |
| --- | --- |
| main | Electron 的入口文件，一般情况下可以自由配置。但如果项目使用 `create-react-app` 脚手架创建，则入口文件必须配置为 `public/electron.js` |
| build.win.extraFiles | 打包 Windows 程序时， `electron-builder` 会把 `from` 所指目录下的所有文件复制到 bin/win-unpacked/resources（全小写） |
| build.mac.extraFiles | 打包 Mac 程序时， `electron-builder` 会把 `from` 指向的 `trtc_electron_sdk.node` 文件复制到 bin/mac/your-app-name.app/Contents/Resources（首字母大写） |
| build.directories.output | 打包文件的输出路径。例如这个配置会输出到 `bin` 目录下，可根据实际需要修改 |

| | |
|---|---|
| build.scripts.build:mac | 以 Mac 平台为目标构建脚本 |
| build.scripts.build:win | 以 Windows 平台为目标构建脚本 |
| build.scripts.compile:mac | 编译为 Mac 下的 .dmg 安装文件 |
| build.scripts.compile:win64 | 编译为 Windows 下的 .exe 安装文件 |
| build.scripts.pack:mac | 先调用 build:mac 构建代码，再调用 compile:mac 打包成 .dmg 安装文件 |
| build.scripts.pack:win64 | 先调用 build:win 构建代码，再调用 compile:win64 打包成 .exe 安装文件 |

### 步骤4：执行打包命令

- 打包 Mac.dmg 安装文件：

```
$ cd [项目目录]
$ npm run pack:mac
```

成功执行后，打包工具会生成 `bin/your-app-name-0.1.0.dmg` 安装文件，请选择此文件发布。

- 打包 Windows.exe 安装文件：

```
$ cd [项目目录]
$ npm run pack:win64
```

成功执行后，打包工具会生成 `bin/your-app-name Setup 0.1.0.exe` 安装文件，请选择此文件发布。

> 注意：
> TRTC Electron SDK 暂不支持跨平台打包（例如在 Mac 下打包 Windows 的 .exe 文件，或在 Windows 平台下打包 Mac 的 .dmg 文件）。目前我们正在研究跨平台打包方案，敬请期待。

## 常见问题

### 1. 防火墙有什么限制？

由于 SDK 使用 UDP 协议进行音视频传输，所以对 UDP 有拦截的办公网络下无法使用，如遇到类似问题，请参见应对公司防火墙限制。

### 2. Electron 安装或打包异常

- 如果您在集成 Electron 过程中遇到异常：例如安装超时或失败，打包后出现 trtc_electron_sdk.node 文件加载失败等情况，相关问题解答请联系我们。

## 参考文档

- SDK API 手册
- SDK 更新日志
- Simple Demo 源码
- API Example 源码
- Electron 常见问题

# 快速集成(Flutter)

最近更新时间：2022-04-06 17:10:29

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

> 注意：
> 目前Flutter SDK仅支持Android和iOS

## 环境要求

- Flutter 2.0及以上版本。
- Android 端开发：
  - Android Studio 3.5及以上版本。
  - App 要求 Android 4.1及以上版本设备。
- iOS 端开发：
  - Xcode 11.0及以上版本。
  - 请确保您的项目已设置有效的开发者签名。

## 集成 SDK

Flutter SDK 已经发布到 pub 库，您可以通过配置 `pubspec.yaml` 自动下载更新。

1. 在项目的 `pubspec.yaml` 中写如下依赖：

```
dependencies:
tencent_trtc_cloud: 最新版本号
```

2. 开通**摄像头**和**麦克风**的权限，即可开启语音通话功能。

### iOS端

1. 需要在 `Info.plist` 中加入对相机和麦克风的权限申请：

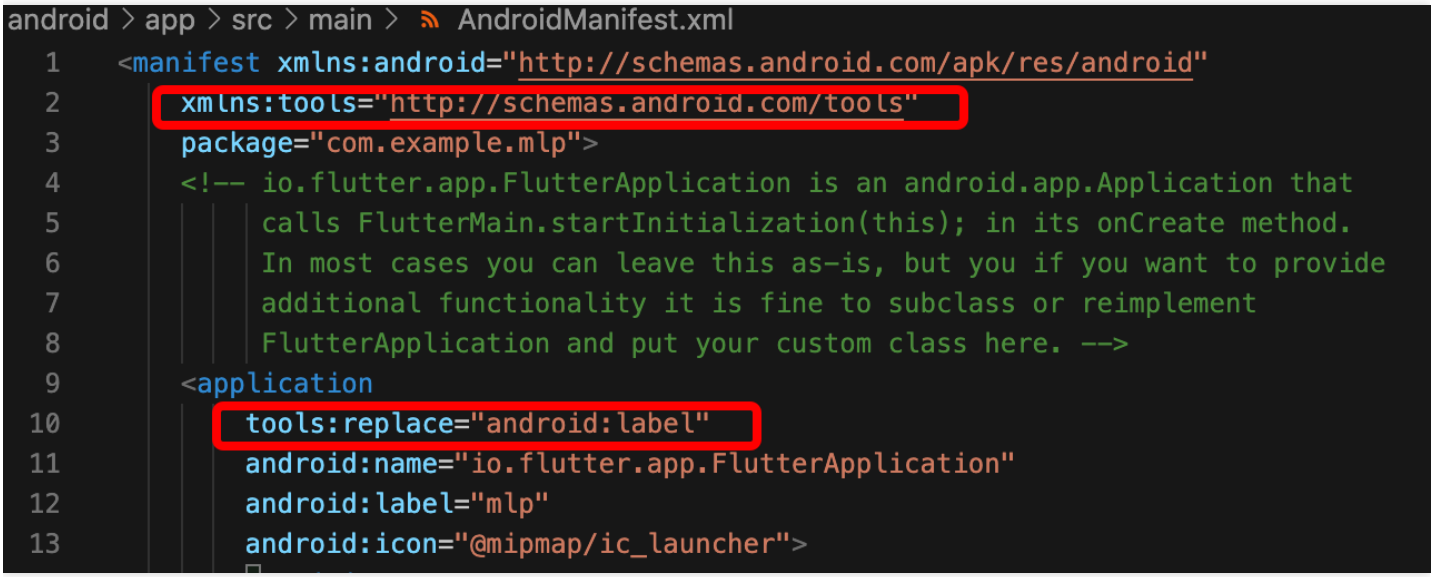
```
<key>NSCameraUsageDescription</key>
<string>授权摄像头权限才能正常视频通话</string>
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

2. 添加字段 `io.flutter.embedded_views_preview`，并设定值为 YES。

### Android端

1. 打开 `/android/app/src/main/AndroidManifest.xml` 文件。
2. 将 `xmlns:tools="http://schemas.android.com/tools"` 加入到 manifest 中。
3. 将 `tools:replace="android:label"` 加入到 application 中。

> 说明：
> 若不执行此步，会出现 Android Manifest merge failed 编译失败 问题。




## 常见问题

- iOS 打包运行 Crash？
- iOS 无法显示视频（Android 正常）？
- 更新 SDK 版本后，iOS CocoaPods 运行报错？
- Android Manifest merge failed 编译失败？

- 因为没有签名，真机调试报错?
- 对插件内的 swift 文件做了增删后，build 时查找不到对应文件？
- Run 报错“Info.plit, error: No value at that key path or invalid key path: NSBonjourServices”？
- Pod install 报错？
- Run 的时候 iOS 版本依赖报错？

# 快速集成(QT)

最近更新时间：2021-07-29 18:37:02

本文主要介绍如何快速地将腾讯云 TRTC SDK（QT 的 Mac 和 Windows 版本）集成到您的项目中，只要按照如下步骤进行配置，就可以快速完成 SDK 的集成工作。

## Mac 端集成

### 开发环境要求

- 操作系统：Mac10.10及以上版本。
- 开发环境：Qt Creator 4.10.3及以上版本，推荐使用 Qt Creator 4.13.3及以上。
- 开发框架：Based on Qt 5.10及以上。

### 操作步骤

本节以从0创建一个简单的 QTTest 项目为例，介绍如何在 Qt Creator 工程中集成 C++ 跨平台 SDK。

#### 步骤1：下载 C++ 跨平台 SDK

1. 下载 SDK，解压并打开文件。
2. 在您的QTTest同级目录下新建一个空的SDK文件夹，将第1步下载的 `TXLiteAVSDKTRTCMacx.x.x/SDK/TXLiteAVSDKTRTC_Mac.framework` 拷贝到与您 QTTest 工程目录同级目录的 SDK 文件夹下。

#### 步骤2：配置 QTTest.pro

打开 QTTest 工程目录，使用一任意文本编辑器打开 `QTTest.pro` 文件，然后添加 SDK 相关引用：

```
INCLUDEPATH += $$PWD/.
DEPENDPATH += $$PWD/.
LIBS += "-F$$PWD/base/util/mac/usersig"
LIBS += "-F$$PWD/../SDK"
LIBS += -framework TXLiteAVSDK_TRTC_Mac
LIBS += -framework Accelerate
LIBS += -framework AudioUnit
INCLUDEPATH += $$PWD/../SDK/TXLiteAVSDK_TRTC_Mac.framework/Headers/cpp_interface
INCLUDEPATH += $$PWD/base/util/mac/usersig/include
DEPENDPATH += $$PWD/base/util/mac/usersig/include
```

#### 步骤3：授权摄像头和麦克风使用权限

因为 SDK 会使用您的摄像头和麦克风，所以您需要在对应的 `Info.plist` 添加对应的权限申请说明：

```
NSMicrophoneUsageDescription : 申请使用麦克风
NSCameraUsageDescription : 申请使用摄像头
```

如下图所示：

#### 步骤4：引用 TRTC SDK

1. 您可以通过头文件 `#include "ITRTCCloud.h"` 直接引用。
2. 使用命名空间：C++ 全平台接口的方法、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间。

> 说明：
> 至此您的集成工作已经完成，可以编译运行您的项目了。关于更多跨平台 SDK 的 API 使用 Demo，请下载 QTDemo 详细参考。

## Windows 端集成

### 开发环境要求

- 操作系统：Windows 7及以上版本。
- 开发环境：Visual Studio 2015及以上版本，推荐使用 Visual Studio 2015，前提是您已经配置好 VS 相关的 QT 开发环境。

> 说明：
> 如果您不熟悉配置 VS 相关 QT 开发环境的步骤，请参见 README 中的第2条内容。

### 操作步骤

本节以创建一个简单的 QTTest 项目为例，介绍如何在 Visual Studio 工程中集成 C++ 跨平台SDK。

#### 步骤1：下载 C++ 跨平台 SDK

1. 下载 SDK，解压并打开文件。
2. 您的 QTTest 同级目录下新建一个空的SDK文件夹，将第1步下载的 `TXLiteAVSDKTRTCWin_latest/SDK/CPlusPlus` 拷贝到与您 QTTest 工程目录同级目录的 SDK 文件夹下。

#### 步骤2：配置 QTTest 工程依赖环境

**场景一：使用QtCreator配置依赖环境**

打开 QTTest工程目录，使用一款任意文本编辑器（推荐 Sublime Text）打开 `QTTest.pro` （使用 Qt Creator 创建）文件，然后添加 SDK 相关引用：

```
INCLUDEPATH += $$PWD/.
$$PWD/../SDK/CPlusPlus/Win32/include ¥
$$PWD/../SDK/CPlusPlus/Win32/include/TRTC
DEPENDPATH += $$PWD/.
$$PWD/../SDK/CPlusPlus/Win32/include ¥
$$PWD/../SDK/CPlusPlus/Win32/include/TRTC
CONFIG += opengl
CONFIG += debug_and_release
debug {
contains(QT_ARCH,i386) {
LIBS += -L$$PWD/../SDK/CPlusPlus/Win32/lib -lliteav
} else {
LIBS += -L$$PWD/../SDK/CPlusPlus/Win64/lib -lliteav
}
}
release {
contains(QT_ARCH,i386) {
LIBS += -L$$PWD/../SDK/CPlusPlus/Win32/lib -lliteav
} else {
LIBS += -L$$PWD/../SDK/CPlusPlus/Win64/lib -lliteav
}
}
```

**场景二：使用VS配置依赖环境**

如果您的工程已经是一个成熟的 VS 项目，您也可以在 VS 中的工程属性 `Properties->Linker->Input 和 General` 配置 SDK 库路径依赖信息，同时在 `Properties -> C/C++ -> General` 设置好 SDK 的头文件路径依赖信息。

### 步骤3：拷贝文件

当使用 VS 打开 `QTTest.pro` 工程并自动生成相关的 `debug/release` 文件夹后，您需要将 `SDK/CPlusPlus/Win32/lib` 下的所有的 `.dll` 文件分别拷贝到工程目录下的 `debug/release` 文件夹下。

### 步骤4：引用 TRTC SDK

1. 您可以通过头文件 `#include ”ITRTCCloud.h”` 直接引用。
2. 使用命名空间：C++ 全平台接口的方法、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间。

说明：
至此您的集成工作已经完成，可以编译运行您的项目了。关于更多跨平台 SDK 的 API 使用 Demo，请下载 QTDemo 详细参考。

# 快速集成(Unity)

最近更新时间：2021-09-29 15:49:30

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

## 环境要求

- Unity 建议版本： 2020.2.1f1c1。
- 目前支持 Android、iOS、Windows、Mac（Mac 还在内测中）平台。
- 需要包含 `Android Build Support`、`iOS Build Support`、`Winodows Build Support` 和 `MacOs Build Support` 模块。
- 其中 iOS 端开发还需要：
  - Xcode 11.0及以上版本。
  - 请确保您的项目已设置有效的开发者签名。

## 集成 SDK

1. 下载 SDK 及配套的 Demo 源码。
2. 解压后，把项目中的 `TRTCUnitySDK/Assets/TRTCSDK/SDK` 文件夹拷贝到您项目中的 Assets 目录下。

## 常见问题

### Android 提示网络权限问题？

请将项目中 `/Assets/Plugins/AndroidManifest.xml` 文件放到同级目录下。

### Android 没有音视频的权限？

Android 端的麦克风、摄像头权限要手动申请，具体方法请参见以下代码：

```
#if PLATFORM_ANDROID
if (!Permission.HasUserAuthorizedPermission(Permission.Microphone))
{
Permission.RequestUserPermission(Permission.Microphone);
}
if (!Permission.HasUserAuthorizedPermission(Permission.Camera))
{
```

```
Permission.RequestUserPermission(Permission.Camera);
}
#endif
```

# 快速集成(React Native)

最近更新时间：2021-12-31 15:28:58

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

## 环境要求

- ReactNative 0.63 及以上版本。
- Node & Watchman，node版本需在 v12 以上
- **Android 端开发：**
  - Android Studio 3.5及以上版本
  - App 要求 Android 4.1及以上版本设备
  - Java Development Kit
- **iOS & macOS 端开发：**
  - Xcode 11.0及以上版本
  - osx 系统版本要求 10.11 及以上版本
  - 请确保您的项目已设置有效的开发者签名
- 环境安装请参见 官方文档

## 集成 SDK

ReactNative SDK 已经发布到 npm，您可以通过配置 `package.json` 安装。

1. 在项目的 `package.json` 中写如下依赖：

```
"dependencies": {
"trtc-react-native": "^2.0.0"
},
```

2. 开通**摄像头**和**麦克风**的权限，即可开启语音通话功能。
   - Android
   - iOS
   i. 在 `AndroidManifest.xml` 中配置 App 的权限，TRTC SDK 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-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

注意
请勿设置 `android:hardwareAccelerated="false"`，关闭硬件加速之后，会导致对方的视频流无法渲染。

ii. Android 端音视频权限需要手动申请。

```
if (Platform.OS === 'android') {
await PermissionsAndroid.requestMultiple([
PermissionsAndroid.PERMISSIONS.RECORD_AUDIO, //音频需要
PermissionsAndroid.PERMISSIONS.CAMERA, // 视频需要
]);
}
```

# 快速集成(Unreal Engine)

最近更新时间：2022-04-02 13:21:06

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

## 环境要求

- 建议Unreal Engine 4.27.1 及以上版本。
- **Android 端开发：**
  - Android Studio版本4.0及以上版本。
  - Visual Studio 2017 15.6版或更高。
  - 只支持真机调试
- **iOS & macOS 端开发：**
  - Xcode 11.0及以上版本。
  - osx 系统版本要求 10.11 及以上版本
  - 请确保您的项目已设置有效的开发者签名。
- **Windows 开发：**
  - 操作系统：Windows 7 SP1 或更高的版本（基于 x86-64 的 64 位操作系统）。
  - 磁盘空间：除安装 IDE 和一些工具之外还应有至少 1.64 GB 的空间。
  - 安装 Visual Studio 2019。

## 集成 SDK

1. 下载 SDK 及配套的 SDK 源码（有疑问可在此处提issue单）。

2. 解压后，把项目中的 `TRTCSDK` 文件夹拷贝到您项目中的 **Source/[project_name]** 目录下，其中 **[project_name]** 表示你项目的名称。

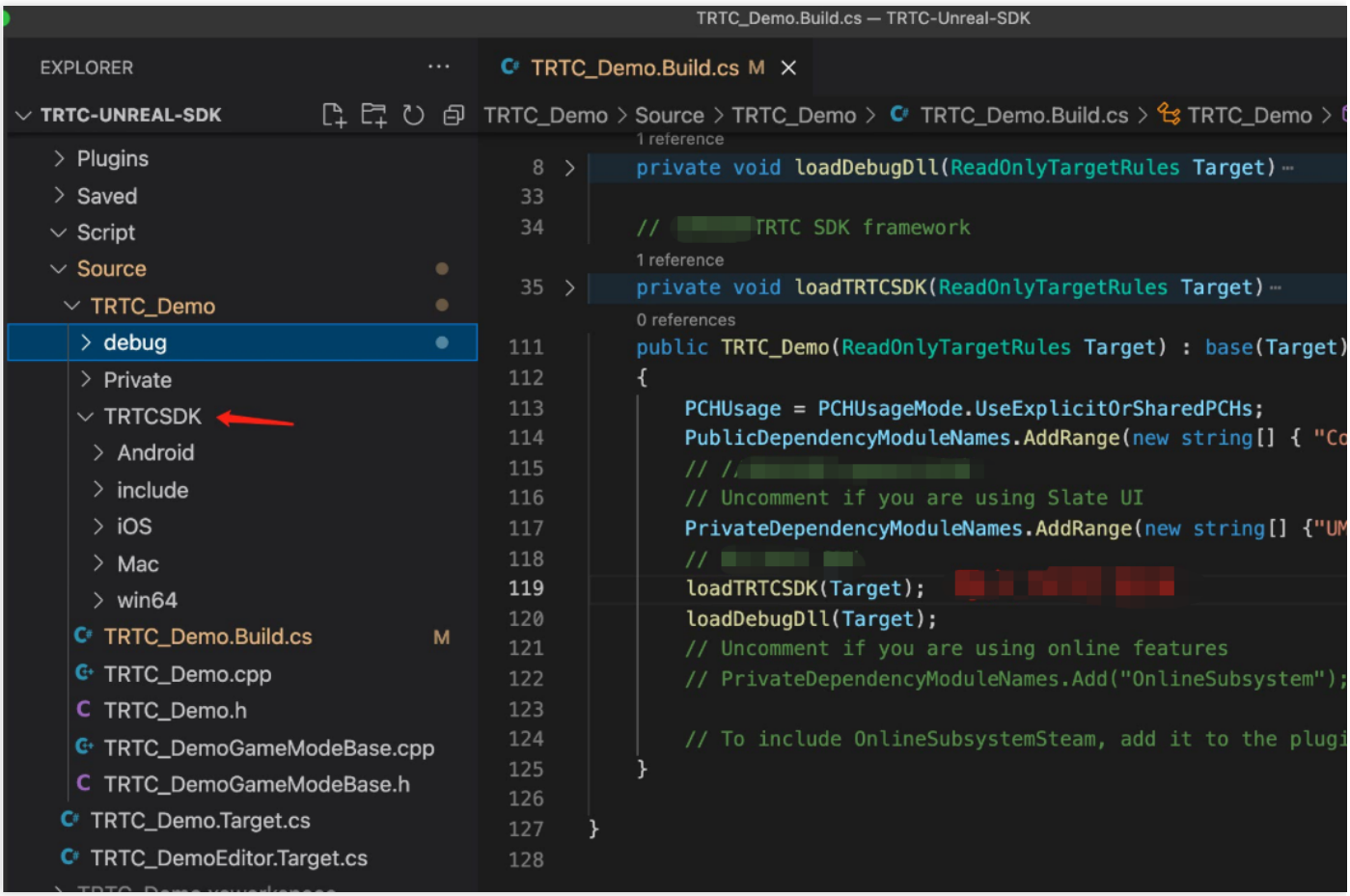
3. 编辑你项目中的 [project_name].Build.cs文件。添加下面函数

```
// 加载各个平台TRTC底层库
private void loadTRTCSDK(ReadOnlyTargetRules Target)
{
string _TRTCSDKPath = Path.GetFullPath(Path.Combine(ModuleDirectory, "TRTCSDK"));
bEnableUndefinedIdentifierWarnings = false;
if (Target.Platform == UnrealTargetPlatform.Android)
```

```
{
// 加载Android头文件
PublicIncludePaths.Add(Path.Combine(_TRTCSDKPath, "include/Android"));
PrivateDependencyModuleNames.AddRange(new string[] { "Launch" });
// 加载Android APL文件
AdditionalPropertiesForReceipt.Add(new ReceiptProperty("AndroidPlugin", Path.Combine(ModuleDir
ectory, "TRTCSDK", "Android", "APL_armv7.xml")));

string Architecture = "armeabi-v7a";
// string Architecture = "arm64-v8a";
// string Architecture = "armeabi";
PublicAdditionalLibraries.Add(Path.Combine(ModuleDirectory,"TRTCSDK", "Android", Architecture,
"libtraeimp-rtmp.so"));
PublicAdditionalLibraries.Add(Path.Combine(ModuleDirectory,"TRTCSDK", "Android", Architecture,
"libliteavsdk.so"));
}else if (Target.Platform == UnrealTargetPlatform.IOS)
{
// 加载iOS头文件
PublicIncludePaths.Add(Path.Combine(_TRTCSDKPath, "include/iOS"));
PublicAdditionalLibraries.AddRange(new string[] {
"resolv",
"z",
"c++",
});
PublicFrameworks.AddRange(
new string[] {
"CoreML",
"VideoToolbox",
"Accelerate",
"CFNetwork",
"OpenGLES",
"AVFoundation",
"CoreTelephony"
}
);
PublicAdditionalFrameworks.Add(new UEBuildFramework( "TXLiteAVSDK_TRTC",_TRTCSDKPath+"/ios/TXL
iteAVSDK_TRTC.framework.zip", ""));
}else if(Target.Platform == UnrealTargetPlatform.Mac)
{
// 加载MacOs头文件
PublicIncludePaths.Add(Path.Combine(_TRTCSDKPath, "include/Mac"));
PublicAdditionalLibraries.AddRange(new string[] {
"resolv",
"z",
"c++",
"bz2",
});
PublicFrameworks.AddRange(
```

```
new string[] {
"AppKit",
"IOKit",
"CoreVideo",
"CFNetwork",
"OpenGl",
"CoreGraphics",
"Accelerate",
"CoreFoundation",
"SystemConfiguration",
"AudioToolbox",
"VideoToolbox",
"CoreTelephony",
"CoreWLAN",
"AVFoundation",
"CoreMedia",
"CoreAudio",
"AudioUnit",
"Accelerate",
});
PublicFrameworks.Add(Path.Combine(_TRTCSDKPath, "Mac", "Release","TXLiteAVSDK_TRTC_Mac.framework"));
}else if (Target.Platform == UnrealTargetPlatform.Win64)
{
// 加载Win64头文件
PublicIncludePaths.Add(Path.Combine(_TRTCSDKPath, "include/win64"));
PublicAdditionalLibraries.Add(Path.Combine(_TRTCSDKPath, "win64", "Release","liteav.lib"));
PublicDelayLoadDLLs.Add(Path.Combine(_TRTCSDKPath, "win64", "Release", "liteav.dll"));
PublicDelayLoadDLLs.Add(Path.Combine(_TRTCSDKPath, "win64", "Release", "LiteAvAudioHook.dll"));
PublicDelayLoadDLLs.Add(Path.Combine(_TRTCSDKPath, "win64", "Release", "LiteAvAudioHookService.dll"));
PublicDelayLoadDLLs.Add(Path.Combine(_TRTCSDKPath, "win64", "Release", "openh264.dll"));
PublicDelayLoadDLLs.Add(Path.Combine(_TRTCSDKPath, "win64", "Release", "TRAE.dll"));
RuntimeDependencies.Add("$(BinaryOutputDir)/liteav.dll", Path.Combine(_TRTCSDKPath, "win64", "Release", "liteav.dll"));
RuntimeDependencies.Add("$(BinaryOutputDir)/LiteAvAudioHook.dll", Path.Combine(_TRTCSDKPath, "win64", "Release", "LiteAvAudioHook.dll"));
RuntimeDependencies.Add("$(BinaryOutputDir)/LiteAvAudioHookService.dll", Path.Combine(_TRTCSDKPath, "win64", "Release", "LiteAvAudioHookService.dll"));
RuntimeDependencies.Add("$(BinaryOutputDir)/openh264.dll", Path.Combine(_TRTCSDKPath, "win64", "Release", "openh264.dll"));
RuntimeDependencies.Add("$(BinaryOutputDir)/TRAE.dll", Path.Combine(_TRTCSDKPath, "win64", "Release", "TRAE.dll"));
}
}
```

4. 在[project_name].Build.cs文件调用该函数




5. 到目前为止你已经集成了TRTC SDK。可在你的cpp 文件中使用TRTC了。 `#include "ITRTCCloud.h"`

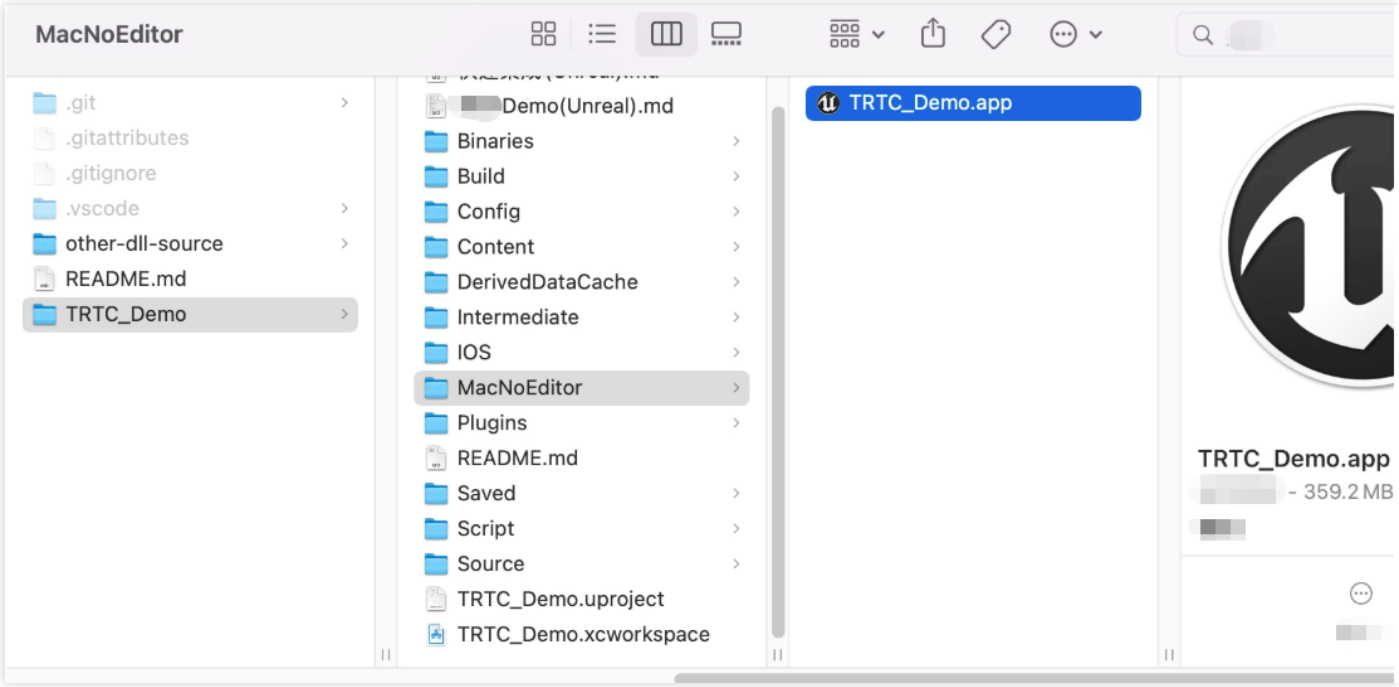
```
// 获取TRTC单例对象
#if PLATFORM_ANDROID
if (JNIEnv* Env = FAndroidApplication::GetJavaEnv()) {
void* activity = (void*) FAndroidApplication::GetGameActivityThis();
// 安卓这里需要传入当前上下文对象
pTRTCCloud = getTRTCShareInstance(activity);
}
#else
pTRTCCloud = getTRTCShareInstance();
#endif
// 注册事件回调
pTRTCCloud->addCallback(this);
// 获取版本号
std::string version = pTRTCCloud->getSDKVersion();
// 进入房间
trtc::TRTCParams params;
params.userId = "123";
```

```
params.roomId = 110;
params.sdkAppId = SDKAppID;
params.userSig = GenerateTestUserSig().genTestUserSig(params.userId, SDKAppID, SECRETKEY);
pTRTCCloud->enterRoom(params, trtc::TRTCAppSceneVideoCall);
```

## 打包

- macOS 端
- Windows 端
- iOS 端
- Android 端

1. File -> Package Project -> Mac
2. 配置权限。右击上一步编译出的 xxx.app 文件 - 选择 "显示包内容"




3. 进入 "Contents->Info.plist"
4. 选择 "Information Property List" 然后添加以下两个权限:

```
<key>NSCameraUsageDescription</key>
<string>授权摄像头权限才能正常视频通话</string>
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

5. 如果你现在在UE4的editor运行的话，需要找到 **UE4Editor.app** 文件并且按照上面步骤添加权限。

# TRTC全平台 C++ API文档

中文文档

英文文档

# 快速跑通示例
# iOS&Mac

最近更新时间：2022-03-30 16:15:22

本文主要介绍如何快速运行腾讯云 TRTC-API-Example（iOS&Mac）。

## 环境要求

- Xcode 11.0及以上版本。
- 请确保您的项目已设置有效的开发者签名。
- Qt Creator 4.13.3（Mac）及以上版本。

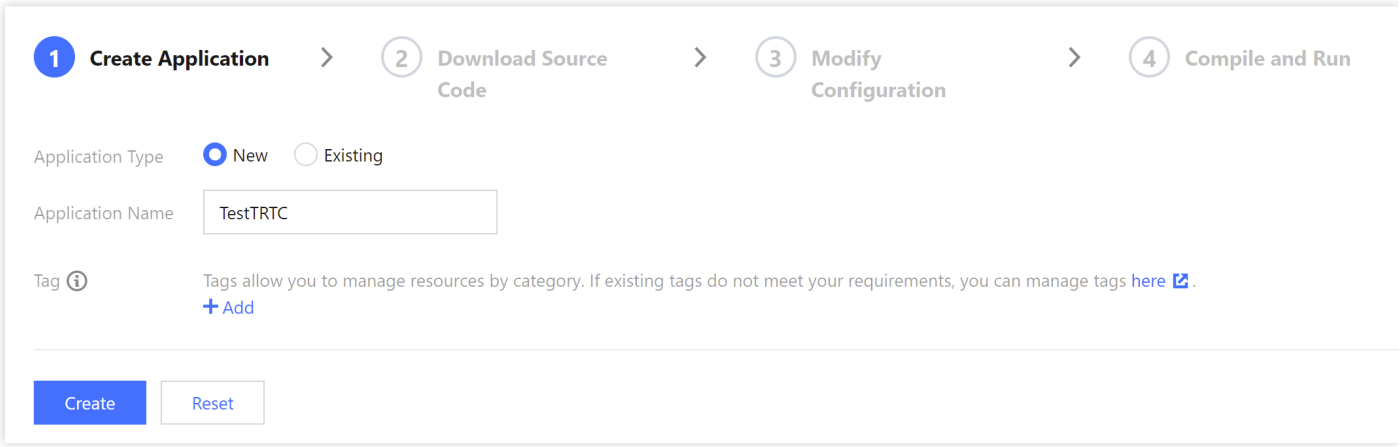
## 前提条件

您已 注册腾讯云 账号，并完成 实名认证。

## 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 单击【新建应用】输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击【选择已有应用】。
3. 根据实际业务需求添加或编辑标签，单击【创建】。

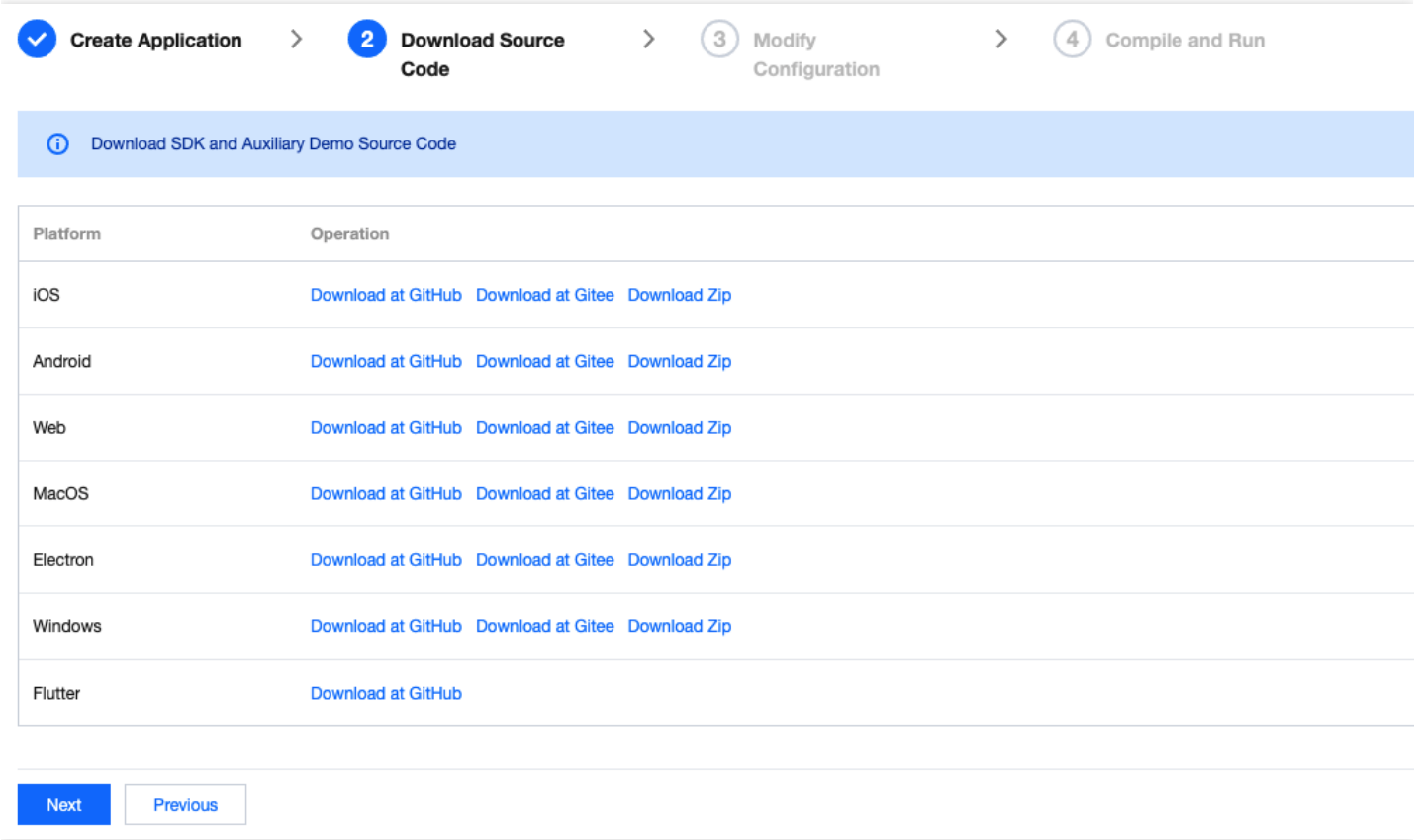


说明：

- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

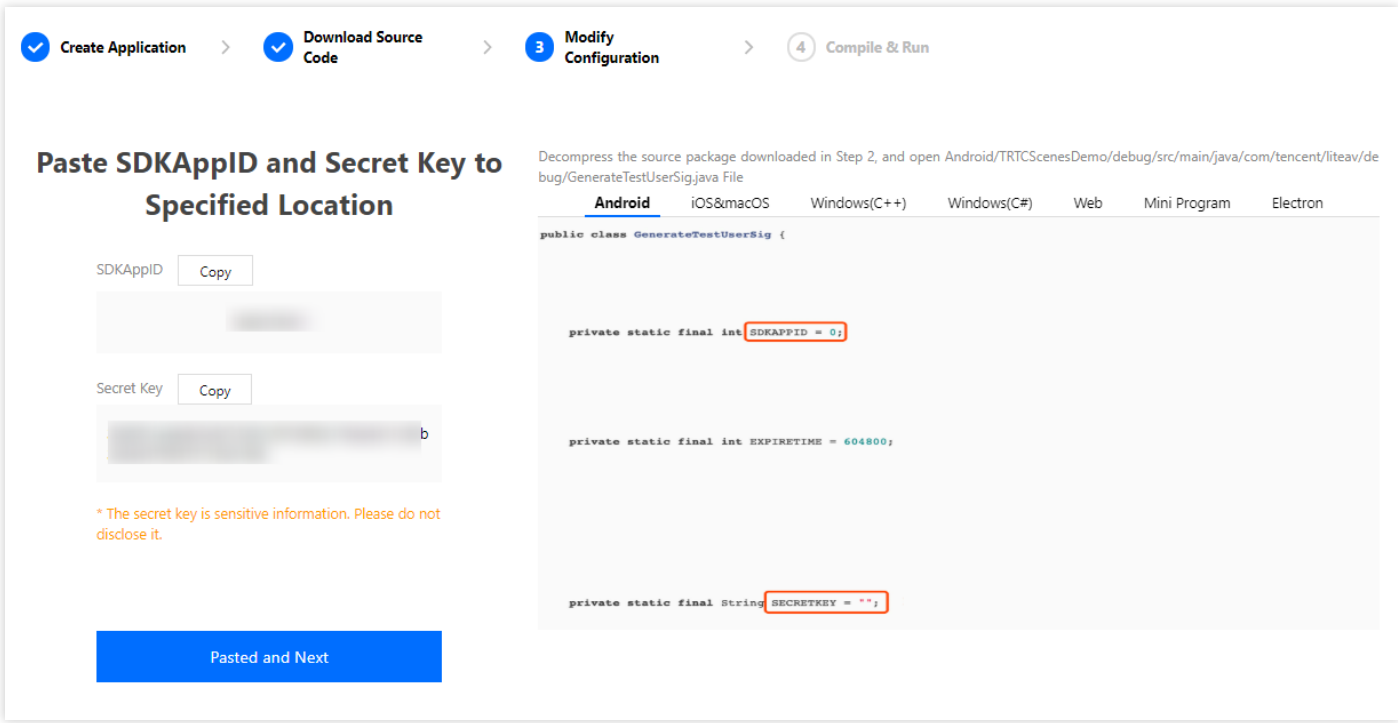
### 步骤2：下载 SDK 和 TRTC-API-Example 源码

1. 根据实际业务需求下载 SDK 及配套的 TRTC-API-Example 源码。
2. 下载完成后，单击【已下载，下一步】。




### 步骤3：配置 TRTC-API-Example 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `LiteAVSDK_TRTC_iOS_版本号/TRTC-API-Example/Debug/GenerateTestUserSig.h` 文件。
3. 设置 `GenerateTestUserSig.h` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。




4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。
5. 编译完成后，单击【回到控制台概览】即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 TRTC-API-Example 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

### 步骤4：编译运行

使用 XCode（11.0及以上的版本）打开源码目录下的 TRTC-API-Example-OC.xcworkspace 工程，编译并运行 TRTC-API-Example 工程即可。

## 常见问题

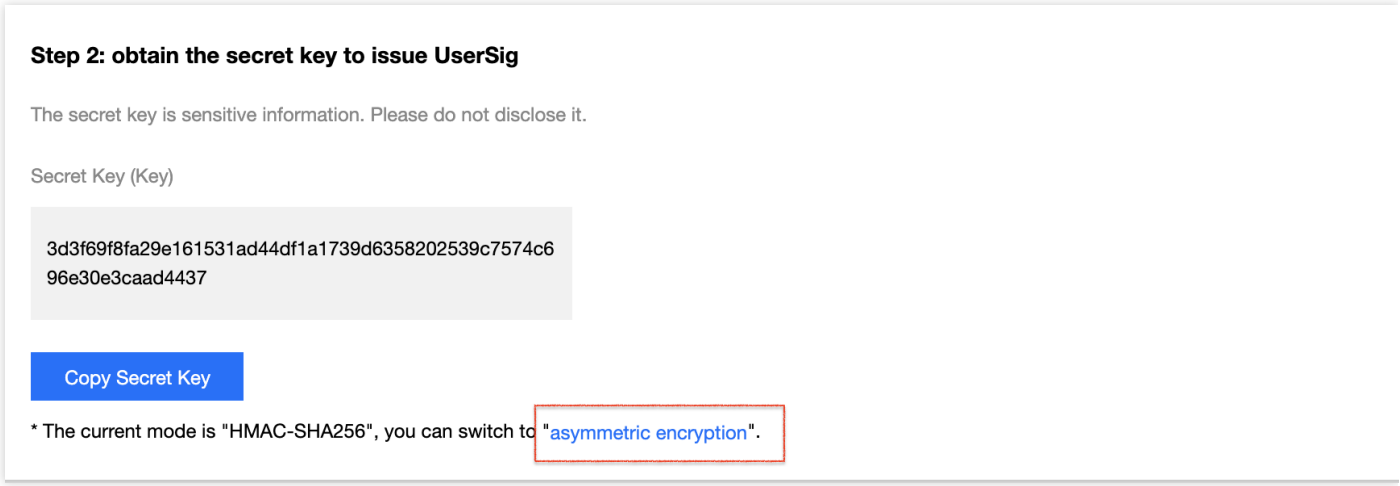
### 1. 查看密钥时只能获取公钥和私钥信息，要如何获取密钥？

TRTC SDK 6.6 版本（2019年08月）开始启用新的签名算法 HMAC-SHA256。在此之前已创建的应用，需要先升级签名算法才能获取新的加密密钥。如不升级，您也可以继续使用 老版本算法 ECDSA-SHA256，如已升级，您按需切换为新旧算法。

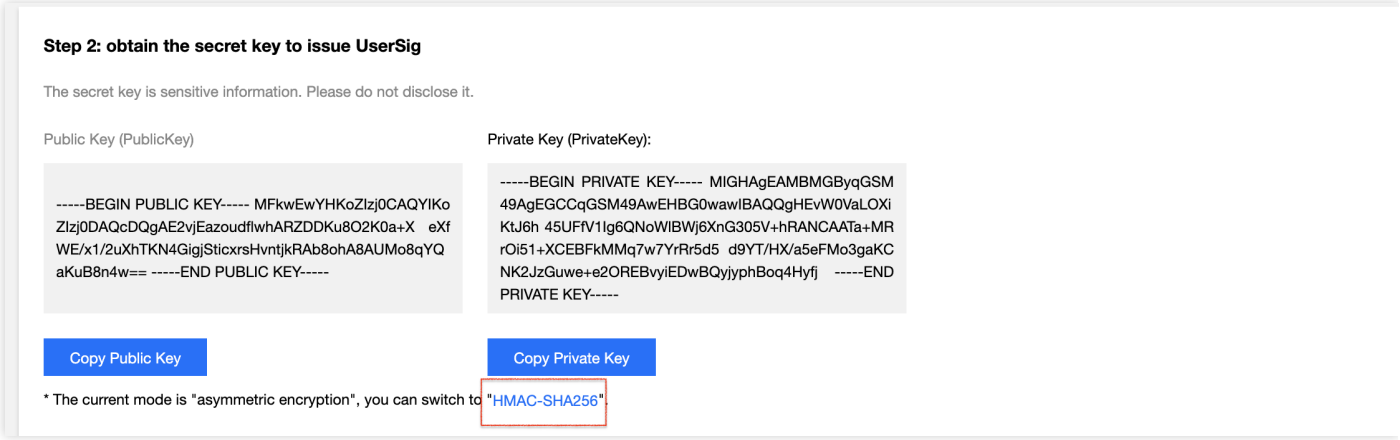
升级/切换操作：

1. 登录 实时音视频控制台。
2. 在左侧导航栏选择【应用管理】，单击目标应用所在行的【应用信息】。
3. 选择【快速上手】页签，单击【第二步 获取签发UserSig的密钥】区域的【点此升级】、【非对称式加密】或【HMAC-SHA256】。

- 升级：
- 切换回老版本算法 ECDSA-SHA256：




- 切换为新版本算法 HMAC-SHA256：




## 2. 两台手机同时运行工程，为什么看不到彼此的画面？

请确保两台手机在运行工程时使用的是不同的 UserID，TRTC 不支持同一个 UserID （除非 SDKAppID 不同）在两个终端同时使用。

### 3. 防火墙有什么限制？

由于 SDK 使用 UDP 协议进行音视频传输，所以在对 UDP 有拦截的办公网络下无法使用。如遇到类似问题，请参见 应对公司防火墙限制 排查并解决。

# Android

最近更新时间：2022-03-30 16:19:37

本文主要介绍如何快速运行腾讯云 TRTC-API-Example（Android）。

## 环境要求

- 最低兼容 Android 4.1（SDK API Level 16），建议使用 Android 5.0 （SDK API Level 21）及以上版本。
- Android Studio 3.5 及以上版本。
- App 要求 Android 4.1 及以上设备。

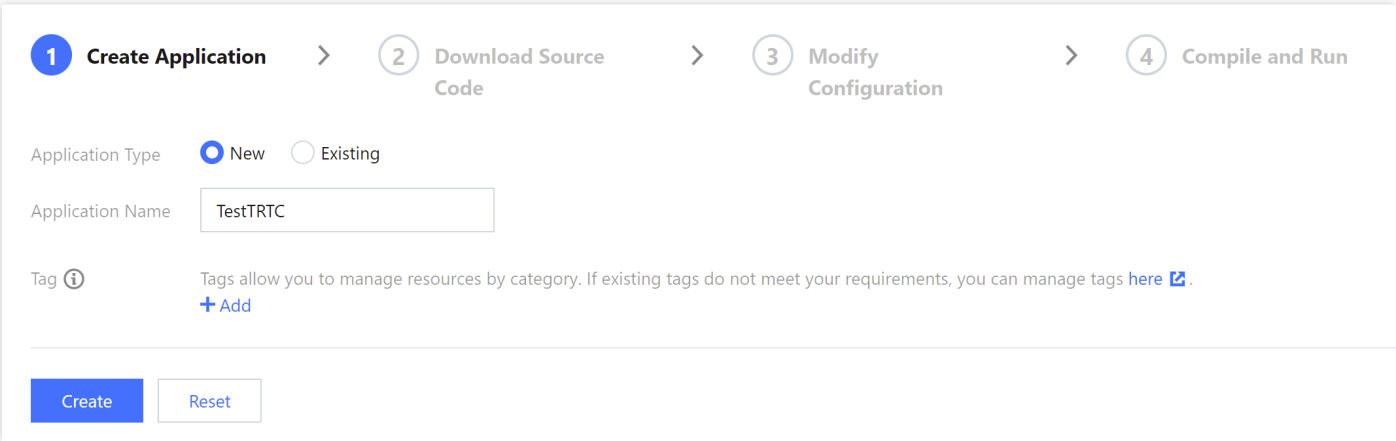
## 前提条件

您已 注册腾讯云 账号，并完成 实名认证。

## 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 单击【新建应用】输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击【选择已有应用】。
3. 根据实际业务需求添加或编辑标签，单击【创建】。

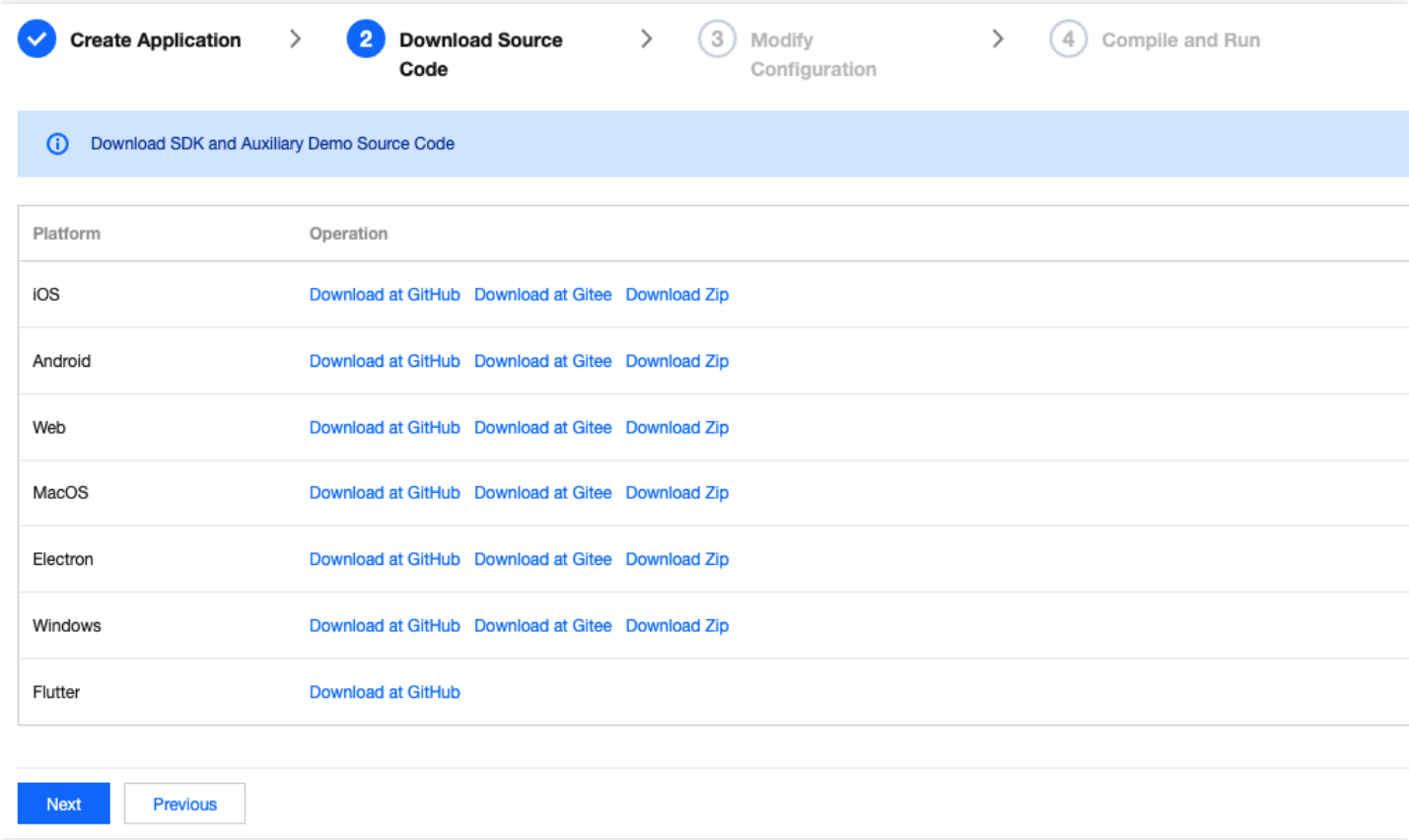



说明：

- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

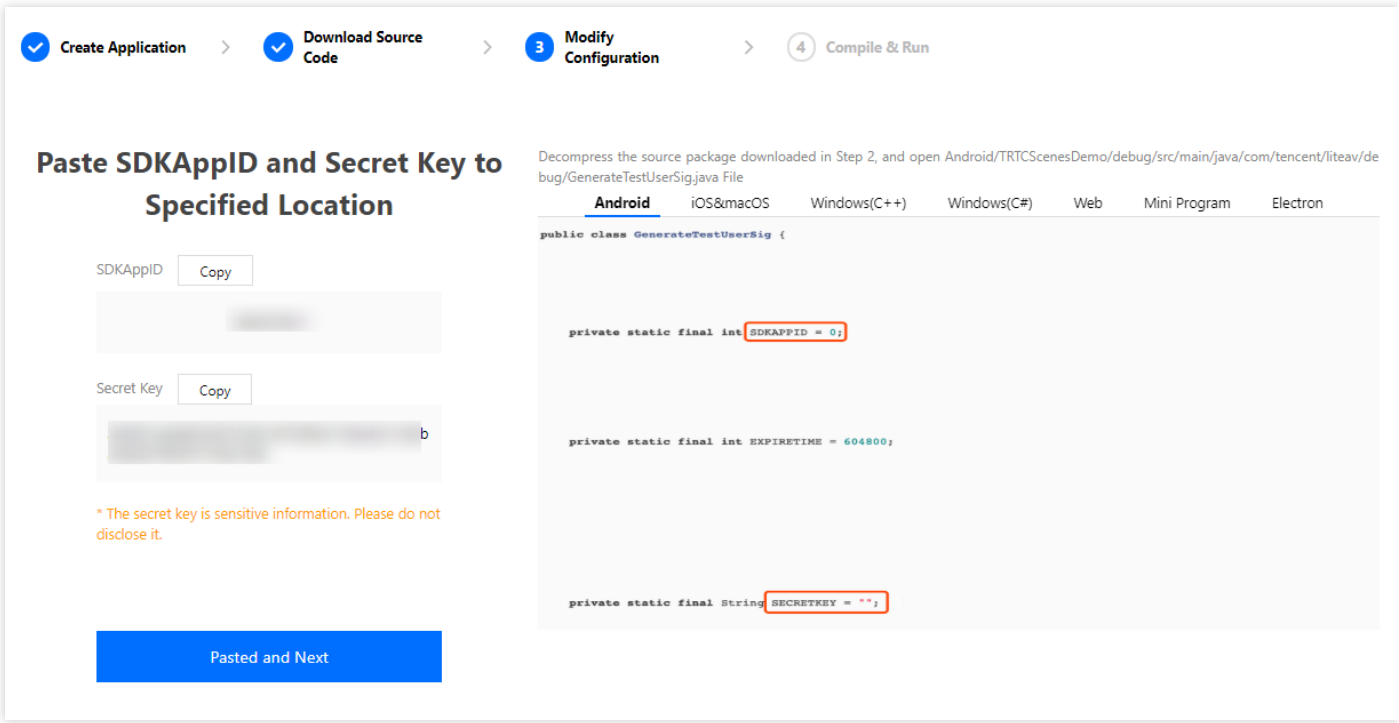
### 步骤2：下载 SDK 和 TRTC-API-Example 源码

1. 根据实际业务需求下载 SDK 及 TRTC-API-Example 源码。
2. 下载完成后，单击【已下载，下一步】。




### 步骤3：配置 TRTC-API-Example 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `LiteAVSDK_TRTC_Android版本号/TRTC-API-Example/Debug/src/main/java/com/tencent/trtc/debug/GenerateTestUserSig.java` 文件。
3. 设置 `GenerateTestUserSig.java` 文件中的相关参数：
   - SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为 PLACEHOLDER，请设置为实际的密钥信息。




4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。
5. 编译完成后，单击【回到控制台概览】即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 TRTC-API-Example 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

### 步骤4：编译运行

使用 Android Studio（3.5及以上的版本）打开源码工程 `TRTC-API-Example`，单击【运行】即可。

# 常见问题

### 1. 查看密钥时只能获取公钥和私钥信息，该如何获取密钥？

TRTC SDK 6.6 版本（2019年08月）开始启用新的签名算法 HMAC-SHA256。在此之前已创建的应用，需要先升级签名算法才能获取新的加密密钥。如不升级，您也可以继续使用 老版本算法 ECDSA-SHA256，如已升级，您按

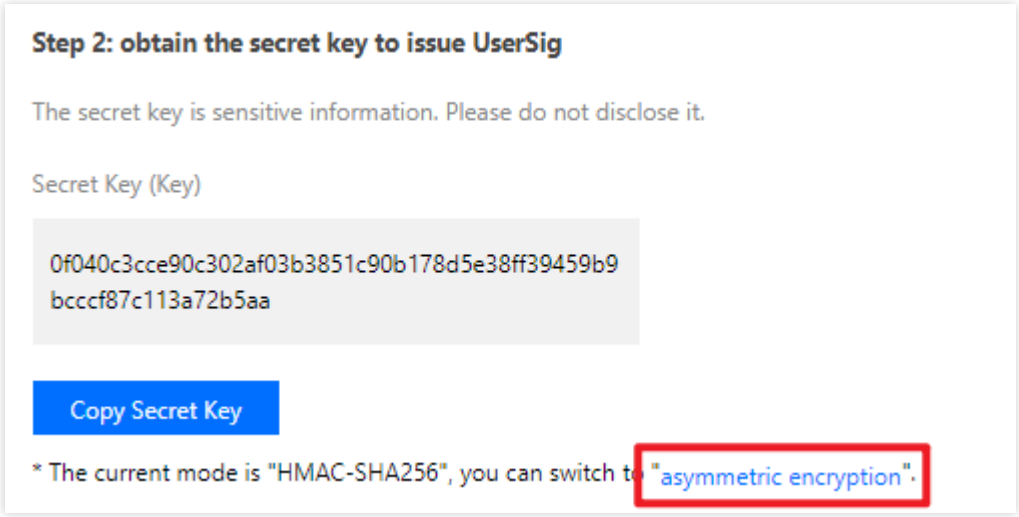
需切换为新旧算法。

升级/切换操作：

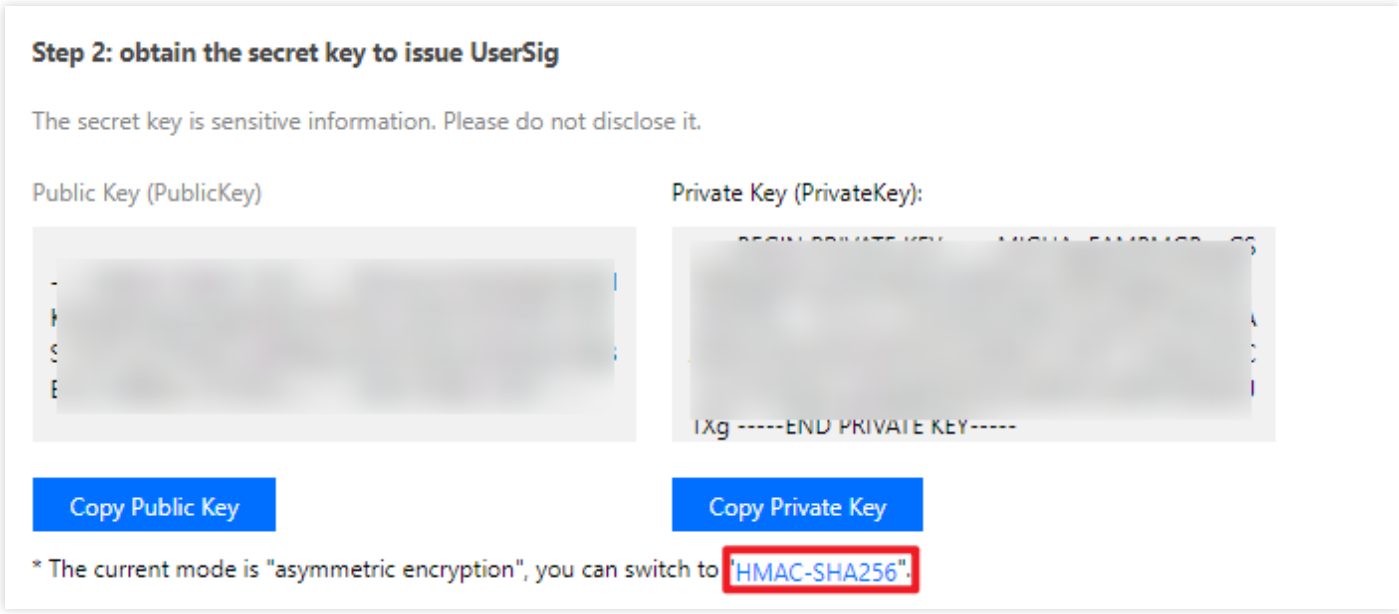
1. 登录 实时音视频控制台。
2. 在左侧导航栏选择【应用管理】，单击目标应用所在行的【应用信息】。
3. 选择【快速上手】页签，单击【第二步 获取签发UserSig的密钥】区域的【点此升级】、【非对称式加密】或【HMAC-SHA256】。

- 升级：

- 切换回老版本算法 ECDSA-SHA256：



- 切换为新版本算法 HMAC-SHA256：




### 2. 两台手机同时运行 App，为什么看不到彼此的画面？

请确保两台手机在运行 App 时使用的是不同的 UserID，TRTC 不支持同一个 UserID （除非 SDKAppID 不同）在两个终端同时使用。

### 3. 防火墙有什么限制？

由于 SDK 使用 UDP 协议进行音视频传输，所以在对 UDP 有拦截的办公网络下无法使用。如遇到类似问题，请参见 应对公司防火墙限制 排查并解决。

# Windows

最近更新时间：2022-03-30 16:23:34

本文主要介绍如何快速运行腾讯云 TRTC Demo（Windows）。

## 环境要求

### Windows（C++）开发环境

- Microsoft Visual Studio 2015及以上版本，推荐使用 Microsoft Visual Studio 2015
- Windows SDK 8.0及以上版本，推荐使用 Windows SDK 8.1

### Windows（C#）开发环境

- Microsoft Visual Studio 2015及以上版本，推荐使用 Microsoft Visual Studio 2017
- .Net Framework 4.0及以上版本，推荐使用 .Net Framework 4.0

### Windows（QT）开发环境

- Microsoft Visual Studio 2015 及以上版本，推荐使用 Microsoft Visual Studio 2015。
- 下载并安装 .vsix 插件文件，官网上找对应插件版本安装即可。
- 打开 VS 并在工具栏找到 `QT VS Tools -> Qt Options -> Qt Versions`，add 添加我们自己的 Qt 编译器 msvc。
- 需要将 `SDK/CPlusPlus/Win32/lib` 下的所有的 `.dll` 文件拷贝到工程目录下的 `debug` / `release` 文件夹下。

> 注意：
> `debug/release` 文件夹均是在 VS 上的环境配置完后自动生成。

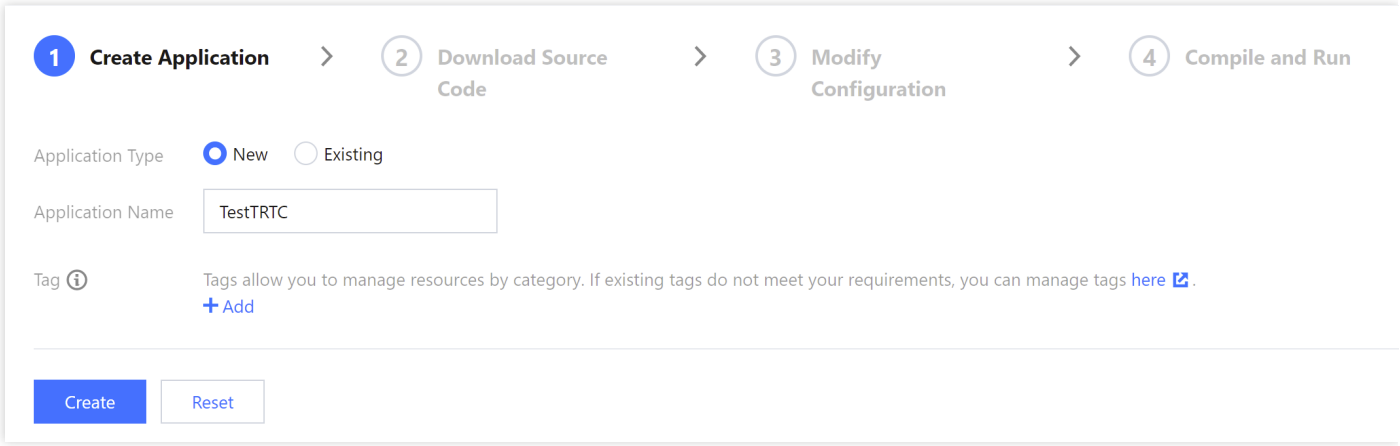
## 前提条件

您已 注册腾讯云 账号，并完成 实名认证。

## 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 单击【新建应用】输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击【选择已有应用】。
3. 根据实际业务需求添加或编辑标签，单击【创建】。



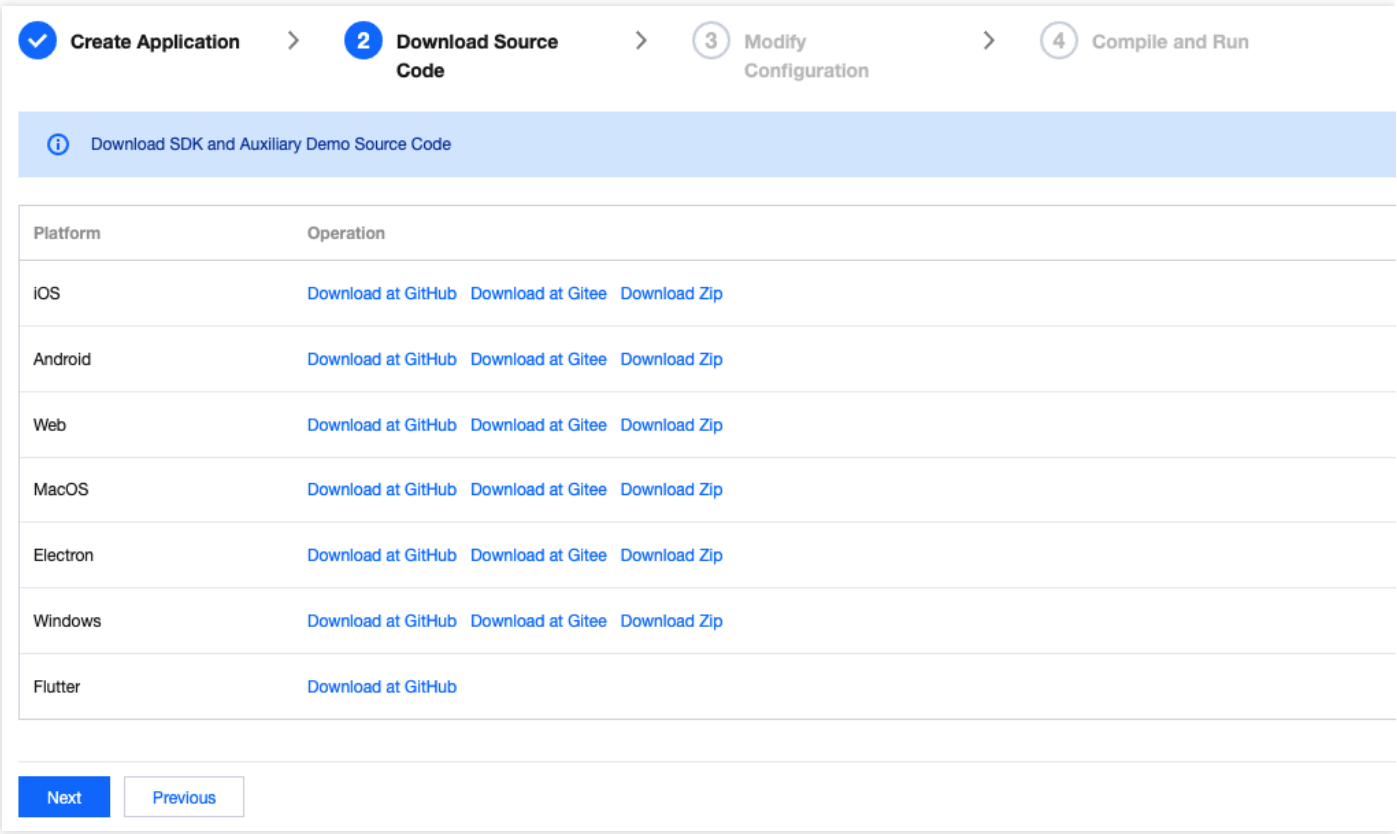

说明：

- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

### 步骤2：下载 SDK 和 Demo 源码

1. 根据实际业务需求下载 SDK 及配套的 Demo 源码。

2. 下载完成后，单击【已下载，下一步】。

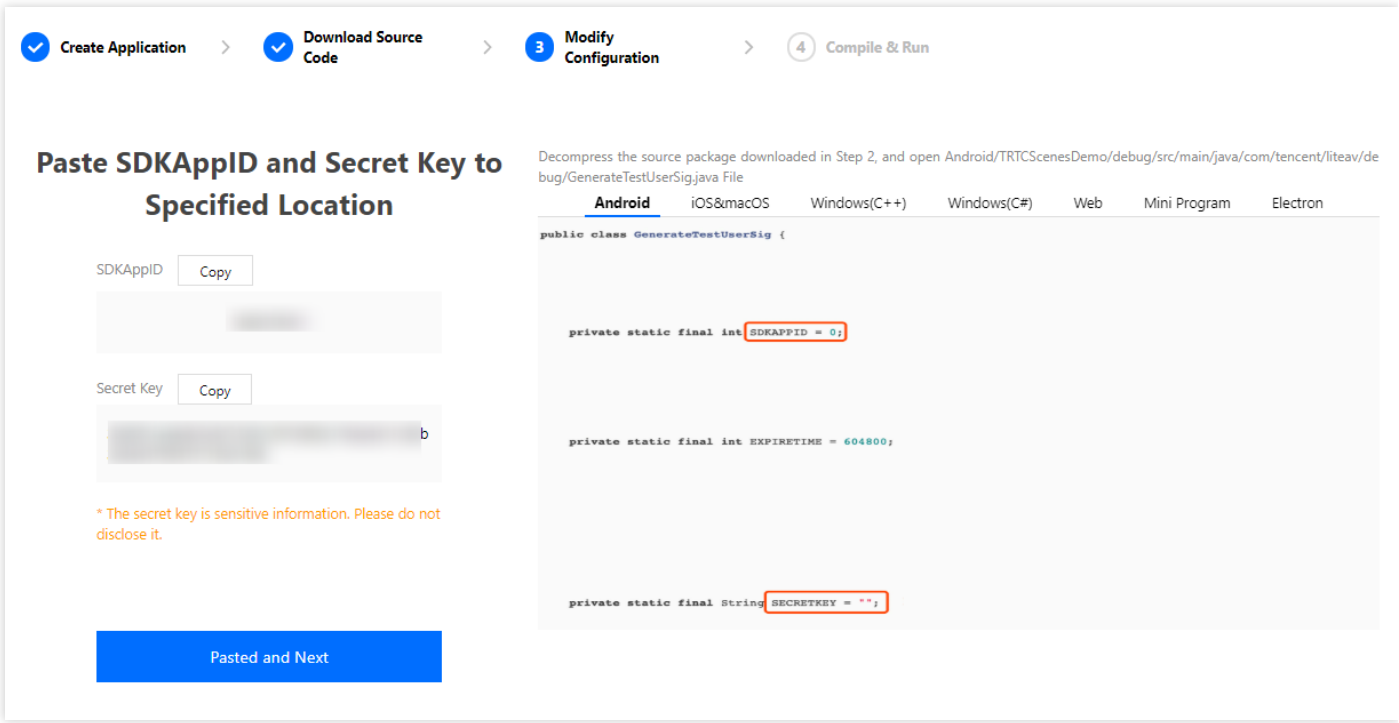



### 步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `GenerateTestUserSig` 文件：

| 适用平台 | 文件相对路径 |
| --- | --- |
| Windows(C++) | Windows/DuilibDemo/GenerateTestUserSig.h |
| Windows(C#) | Windows/CSharpDemo/GenerateTestUserSig.cs |

3. 设置 `GenerateTestUserSig.js` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。




4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。
5. 编译完成后，单击【回到控制台概览】即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

### 步骤4：编译运行

- **Windows（C++）：**
  使用 Visual Studio（建议 VS2015）打开源码目录下的 `DuilibDemo¥TRTCDuilibDemo.sln` 工程文件，推荐选择 Release/X86 构建平台，编译并运行 Demo 工程即可。
- **Windows（C#）：**
  使用 Visual Studio（建议 VS2017）打开源码目录下的 `CSharpDemo¥TRTCCSharpDemo.sln` 工程文件，推荐选择 Release/X86 构建平台，编译并运行 Demo 工程即可。
- **Windows（QT）：**
  使用 Visual Studio（建议 VS2015 或以上）打开源码目录下的

QTDemo\QTDemo.pro工程文件，编译并运行 QTDemo 工程即可。

# 常见问题

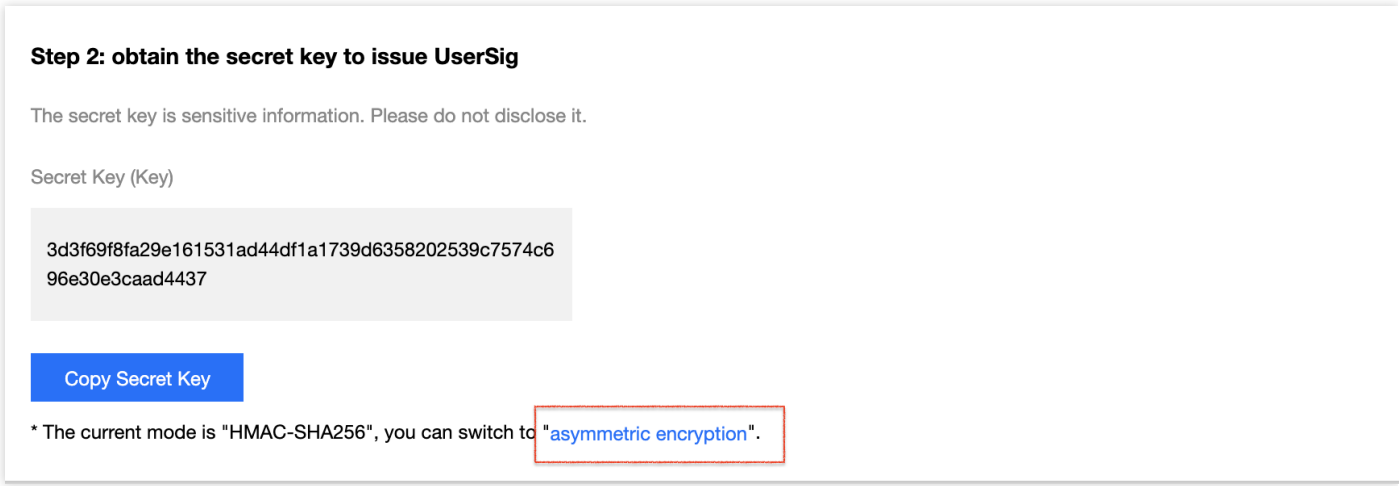
### 1. 查看密钥时只能获取公钥和私钥信息，要如何获取密钥？

TRTC SDK 6.6 版本（2019年08月）开始启用新的签名算法 HMAC-SHA256。在此之前已创建的应用，需要先升级签名算法才能获取新的加密密钥。如不升级，您也可以继续使用 老版本算法 ECDSA-SHA256，如已升级，您按需切换为新旧算法。

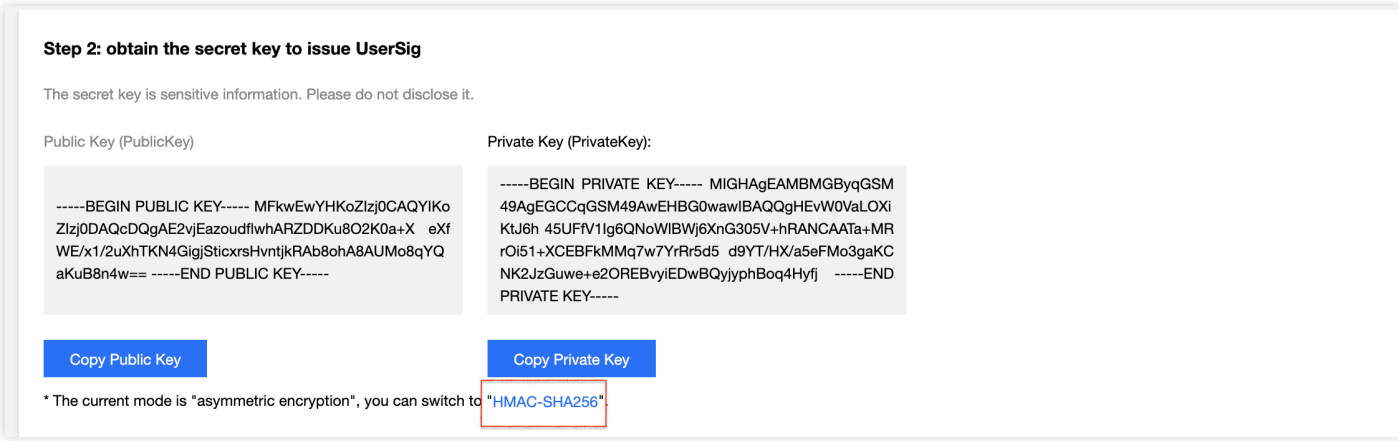
升级/切换操作：

1. 登录 实时音视频控制台。
2. 在左侧导航栏选择【应用管理】，单击目标应用所在行的【应用信息】。
3. 选择【快速上手】页签，单击【第二步 获取签发UserSig的密钥】区域的【点此升级】、【非对称式加密】或【HMAC-SHA256】。

- 升级：
- 切换回老版本算法 ECDSA-SHA256：



- 切换为新版本算法 HMAC-SHA256：




## 2. 两台设备同时运行 Demo，为什么看不到彼此的画面？

请确保两台设备在运行 Demo 时使用的是不同的 UserID，TRTC 不支持同一个 UserID （除非 SDKAppID 不同）在两个设备同时使用。

## 3. 防火墙有什么限制？

由于 SDK 使用 UDP 协议进行音视频传输，所以对 UDP 有拦截的办公网络下无法使用，如遇到类似问题，请参见应对公司防火墙限制。

# Web

最近更新时间：2022-04-15 15:41:05

本文主要介绍如何快速运行腾讯云 TRTC Web SDK Demo。

## 准备工作

运行 TRTC Web SDK Demo 之前需要了解的事项。

### 支持的平台

TRTC Web SDK 基于 WebRTC 实现，目前支持桌面端和移动端的主流浏览器，详细支持度表格请参见 支持的平台。
如果您的应用场景不在支持的表格里，可以打开 TRTC Web SDK 能力检测页面 检测当前环境是否支持 WebRTC 所有能力，例如 WebView 等环境。

如果您的应用场景主要为教育场景，那么教师端推荐使用 Electron 解决方案，支持大小双路画面，更灵活的屏幕分享方案以及更强大的弱网络恢复能力。

### URL 域名协议限制

由于浏览器安全策略的限制，使用 WebRTC 能力对页面的访问协议有严格的要求，请参照以下表格进行开发和部署应用。

| 应用场景 | 协议 | 接收（播放） | 发送（上麦） | 屏幕分享 | 备注 |
| --- | --- | --- | --- | --- | --- |
| 生产环境 | HTTPS 协议 | 支持 | 支持 | 支持 | **推荐** |
| 生产环境 | HTTP 协议 | 支持 | 不支持 | 不支持 | |
| 本地开发环境 | http://localhost | 支持 | 支持 | 支持 | **推荐** |
| 本地开发环境 | http://127.0.0.1 | 支持 | 支持 | 支持 | |
| 本地开发环境 | http://[本机IP] | 支持 | 不支持 | 不支持 | |
| 本地开发环境 | file:/// | 支持 | 支持 | 支持 | |

### 防火墙限制

TRTC Web SDK 依赖以下端口及域名进行数据传输，请将其加入防火墙白名单。配置完成后，您可以通过访问并体验 官网 Demo 检查配置是否生效。具体请参见 应对防火墙限制相关。

- TCP 端口：8687
- UDP 端口：8000，8080，8800，843，443，16285
- 域名： `*.rtc.qq.com` ， `yun.tim.qq.com`

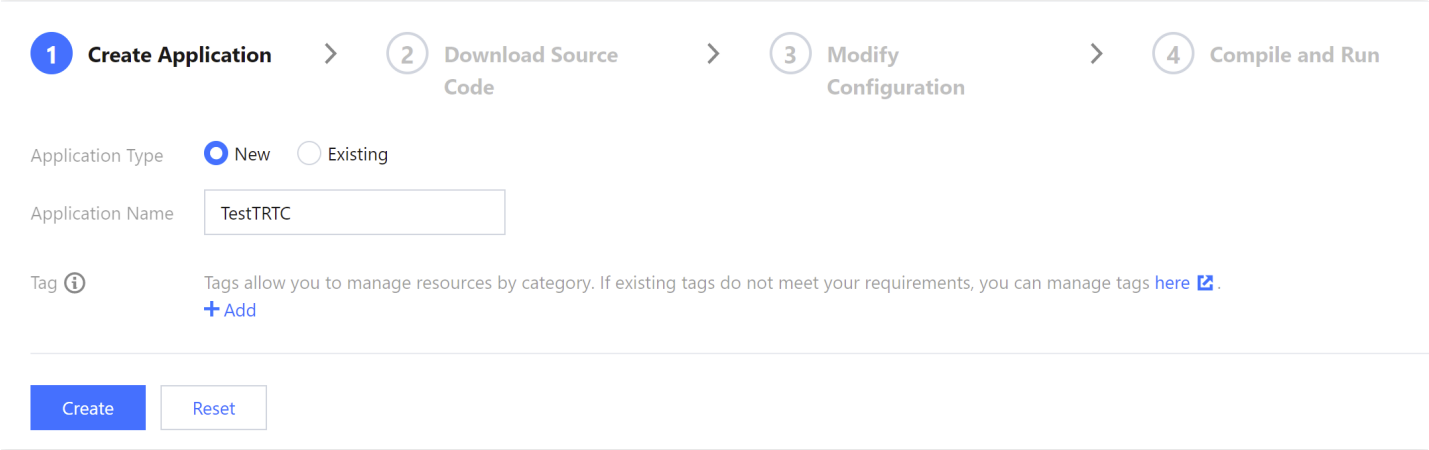
# 前提条件

您已 注册腾讯云 账号，并完成 实名认证。

# 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择 **开发辅助** > **快速跑通Demo**。
2. 单击 **新建应用** 输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击 **选择已有应用**。
3. 根据实际业务需求添加或编辑标签，单击 **创建**。

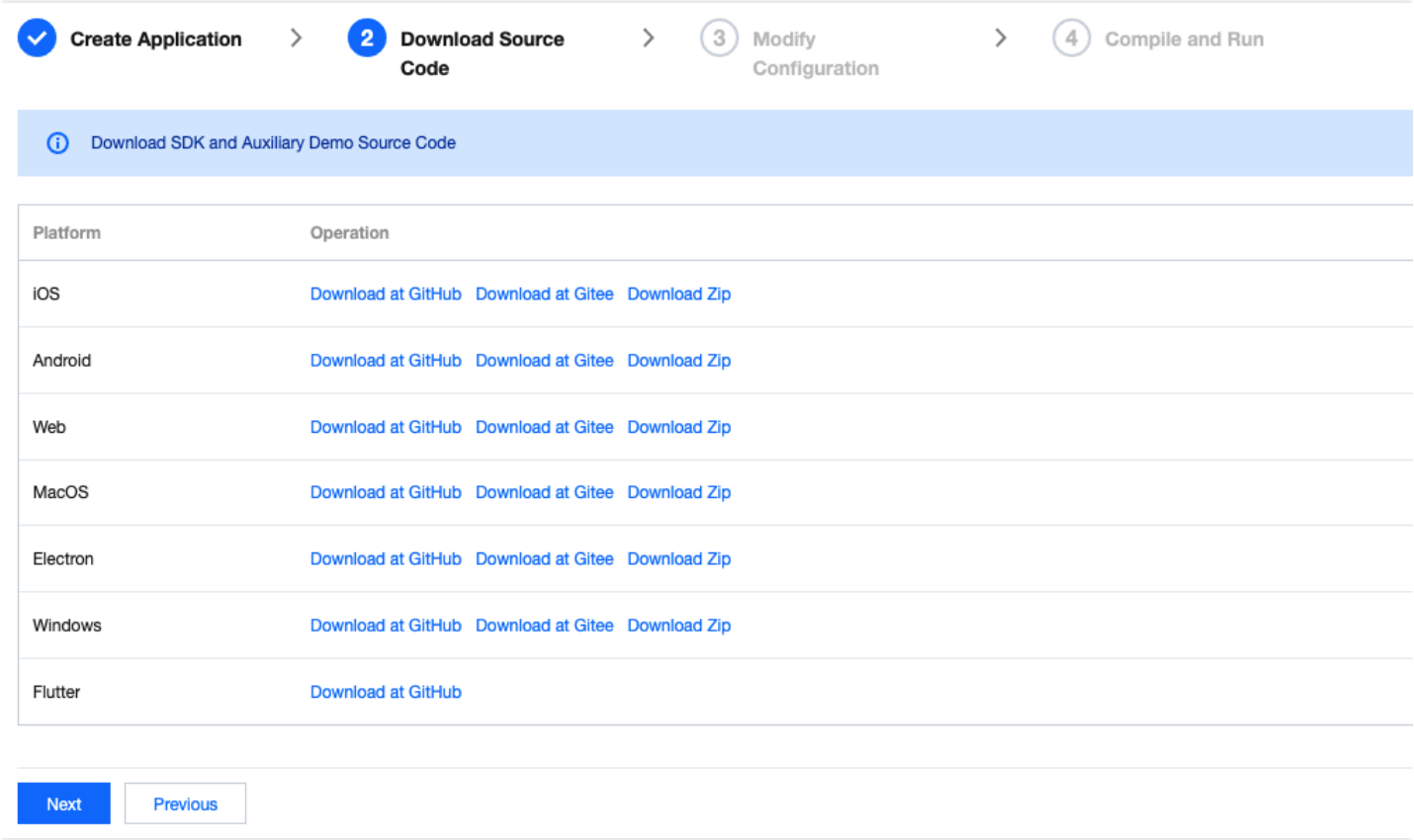



说明：

- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

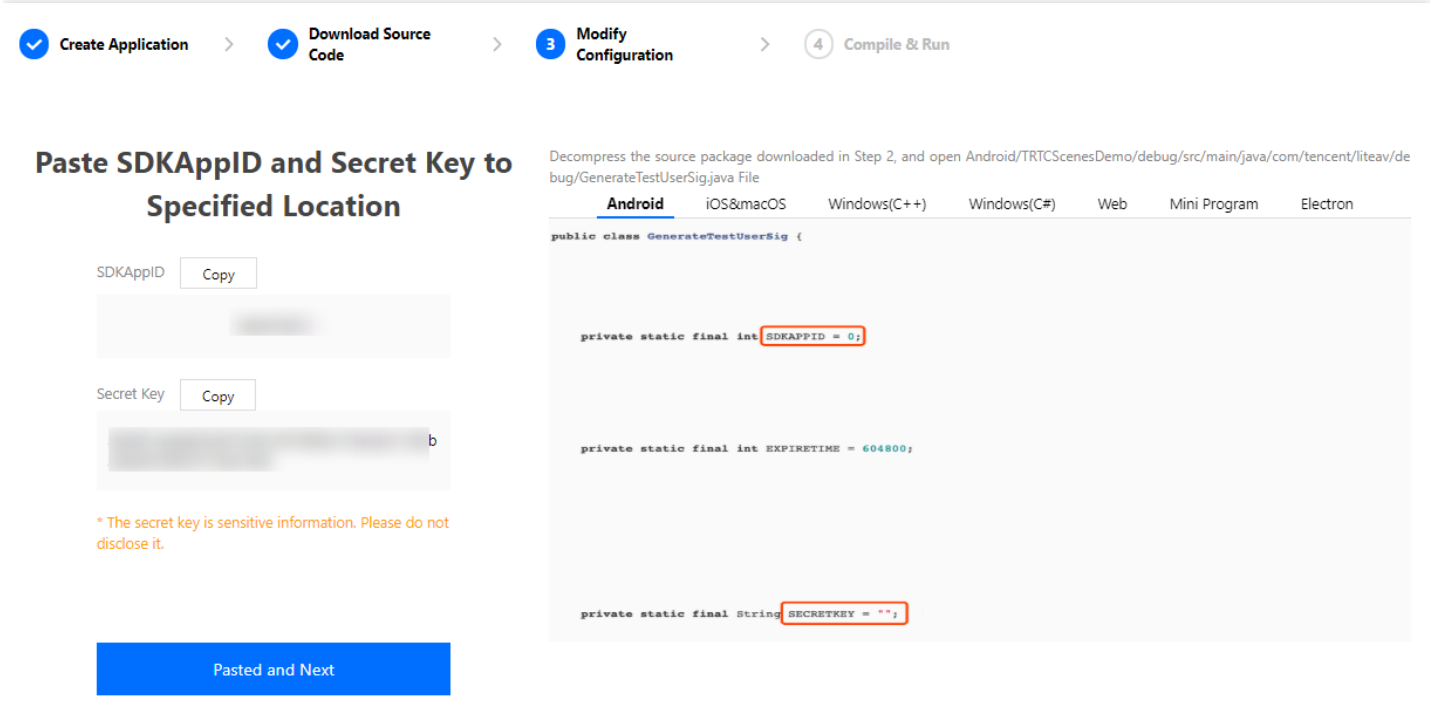
### 步骤2：下载 SDK 和 Demo 源码

1. 下载 Web 端 SDK 及配套的 Demo 源码。
2. 下载完成后，单击 **“已下载，下一步”**。




### 步骤3：获取 SDKAppId 和 密钥（SecretKey）

1. 进入修改配置页，获取 `SDKAppID` 和 `密钥` 。
2. 复制粘贴 SDKAppId 和 密钥（SecretKey）完成后，单击 **已复制粘贴，下一步** 即创建成功。




## 步骤4：运行 Demo

为满足不同客户的需求，TRTC Web 目前提供以下几种基础 Demo：

- `base-js` 为 TRTC Web 基础 Demo。TRTC Web 基础 Demo 集成了 TRTC Web SDK 的基础音视频通话、设备选择等功能，使用 jQuery 开发，可直接在浏览器中运行。快速体验可访问 base-js 在线体验地址。
- `quick-demo-js` 为 TRTC Web 快速运行 Demo (原生 Js 版本)。TRTC Web 快速运行 Demo (原生 Js 版本) 集成了 TRTC Web SDK 的基础音视频通话、设备选择等功能，使用原生 Js 开发，可直接在浏览器中运行。快速体验可访问 quick-demo-js 在线体验地址。
- `quick-demo-vue2-js` 为 TRTC Web 快速运行 Demo (Vue2 版本)。TRTC Web 快速运行 Demo (Vue2 版本) 集成了 TRTC Web SDK 的基础音视频通话、设备选择等功能，使用 Vue2 开发，需要您安装 Node 环境。快速体验可访问 quick-demo-vue2-js 在线体验地址。

- Demo 1：base-js
- Demo 2：quick-demo-js
- Demo 3：quick-demo-vue2-js

1. 在下载的源码中找到并打开 `TRTC_Web/base-js/js/debug/GenerateTestUserSig.js` 文件。
2. 设置 `GenerateTestUserSig.js` 文件中的相关参数：

- SDKAPPID：默认为0，请设置为实际的 `SDKAppID`。
- SECRETKEY：默认为空字符串，请设置为实际的 `密钥` 信息。

3. 运行 Demo：
使用 Chrome 浏览器打开 Demo 根目录下的 `index.html` 文件即可运行 Demo。

> 注意
> - 一般情况下体验 Demo 需要部署至服务器，通过 `https://域名/xxx` 访问，或者直接在本地搭建服务器，通过 `localhost:端口` 访问。
> - 目前桌面端 Chrome 浏览器支持 TRTC Web SDK 的相关特性比较完整，因此建议使用 Chrome 浏览器进行体验。

- 单击 **加入房间** 加入音视频通话房间并且发布本地音视频流。
您可以打开多个页面，每个页面都单击 **加入房间**，正常情况下可以看到多个画面并模拟实时音视频通话。
- 单击摄像头图标可以选择摄像头设备。
- 单击麦克风图标可以选择麦克风设备。

>? WebRTC 需要使用摄像头和麦克风采集音视频，在体验过程中您可能会收到来自 Chrome 浏览器的相关提示，单击 **允许**。

> 注意：
> - 本文使用的生成 UserSig 的方案是在客户端中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

# 常见问题

### 1. 查看密钥时只能获取公钥和私钥信息，要如何获取密钥？

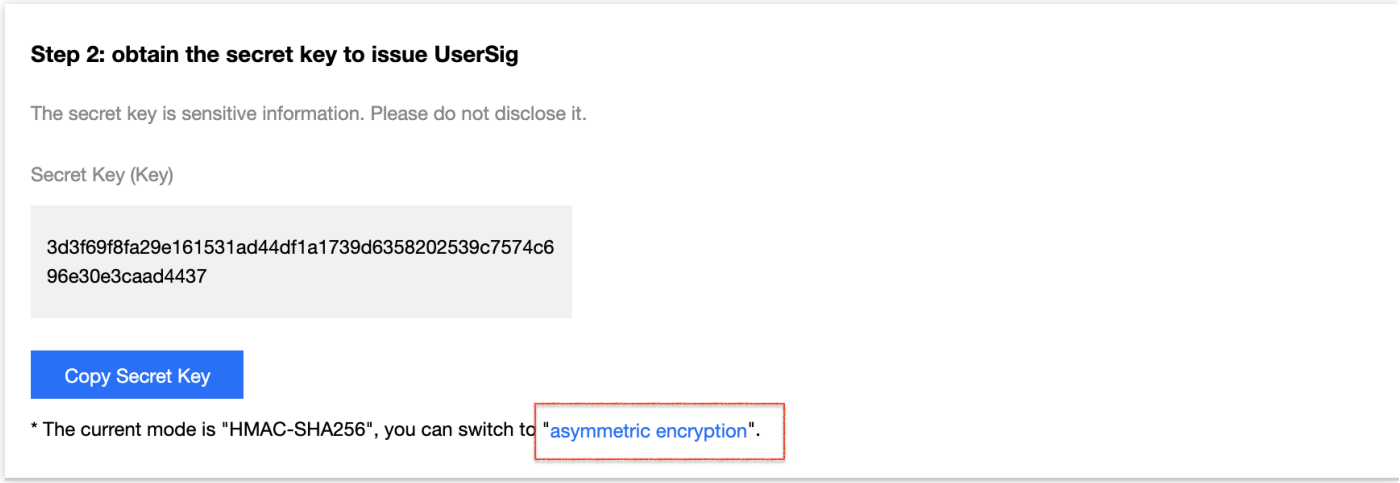
TRTC SDK 6.6（Web SDK 4.0）版本（2019年08月）开始启用新的签名算法 HMAC-SHA256。在此之前已创建的应用，需要先升级签名算法才能获取新的加密密钥。如不升级，您也可以继续使用 老版本算法 ECDSA-SHA256，如已升级，您按需切换为新旧算法。

升级/切换操作：

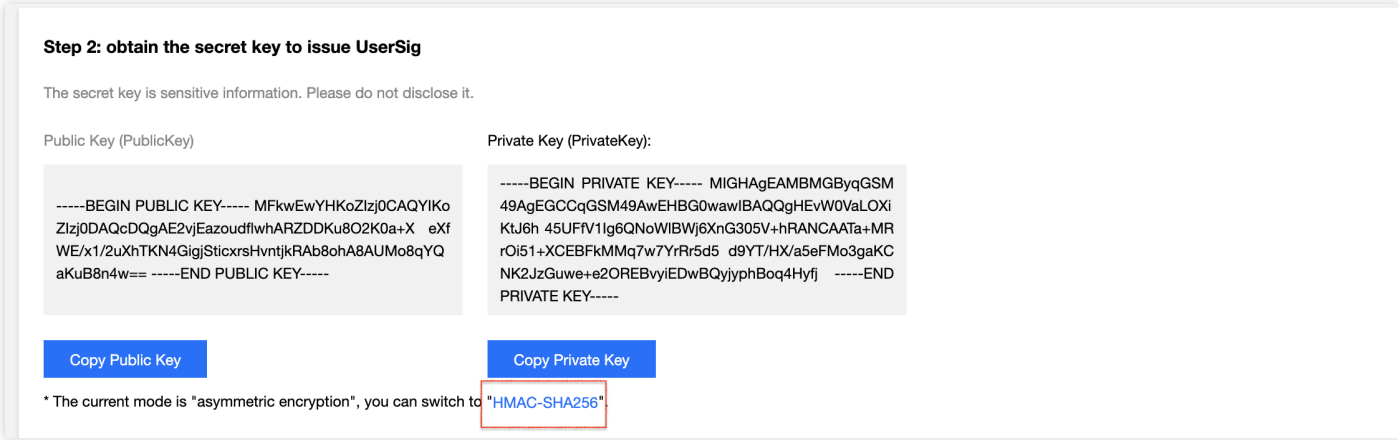
1. 登录 实时音视频控制台。

2. 在左侧导航栏选择 **应用管理**，单击目标应用所在行的 **应用信息**。
3. 选择 **快速上手** 页签，单击 **第二步 获取签发UserSig的密钥** 区域的 **点此升级** 、 **非对称式加密** 或 **HMAC-SHA256**。

- 升级
- 切换回老版本算法 ECDSA-SHA256：




- 切换为新版本算法 HMAC-SHA256：




## 2. 出现客户端错误：“RtcError: no valid ice candidate found”该如何处理？

出现该错误说明 TRTC Web SDK 在 STUN 打洞失败，请根据 环境要求 检查防火墙配置。

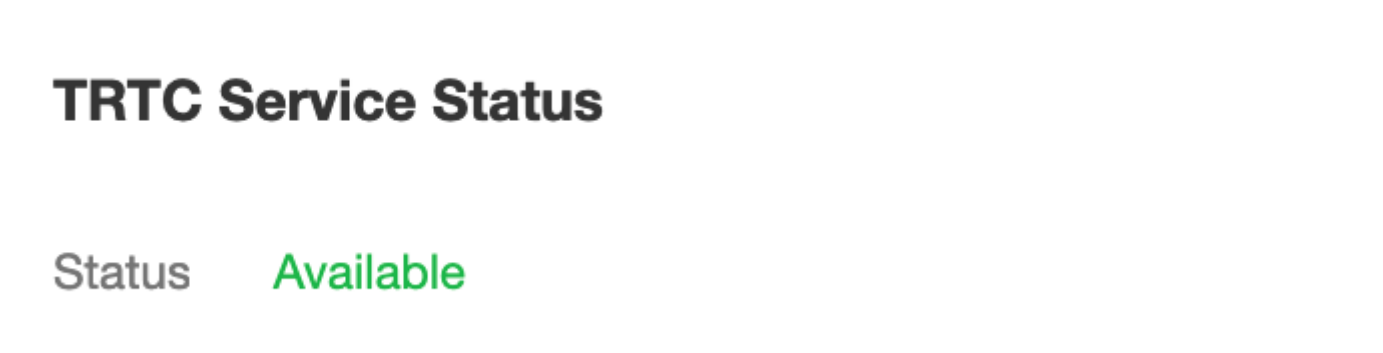
## 3. 出现客户端错误："RtcError: ICE/DTLS Transport connection failed" 或 “RtcError: DTLS Transport connection timeout”该如何处理？

出现该错误说明 TRTC Web SDK 在建立媒体传输通道时失败，请根据 环境要求 检查防火墙配置。

## 4. 出现10006 error 该如何处理？

如果出现"Join room failed result: 10006 error: service is suspended,if charge is overdue,renew it"，请确认您的实时音视频应用的服务状态是否为正常状态。

登录 实时音视频控制台，单击您创建的应用，单击 **帐号信息**，在帐号信息面板即可确认服务状态。




说明：

其他常见问题参见 Web 端相关。

# Electron

最近更新时间：2022-03-30 17:01:13

本文主要介绍如何快速运行腾讯云 TRTC Demo（Electron）。

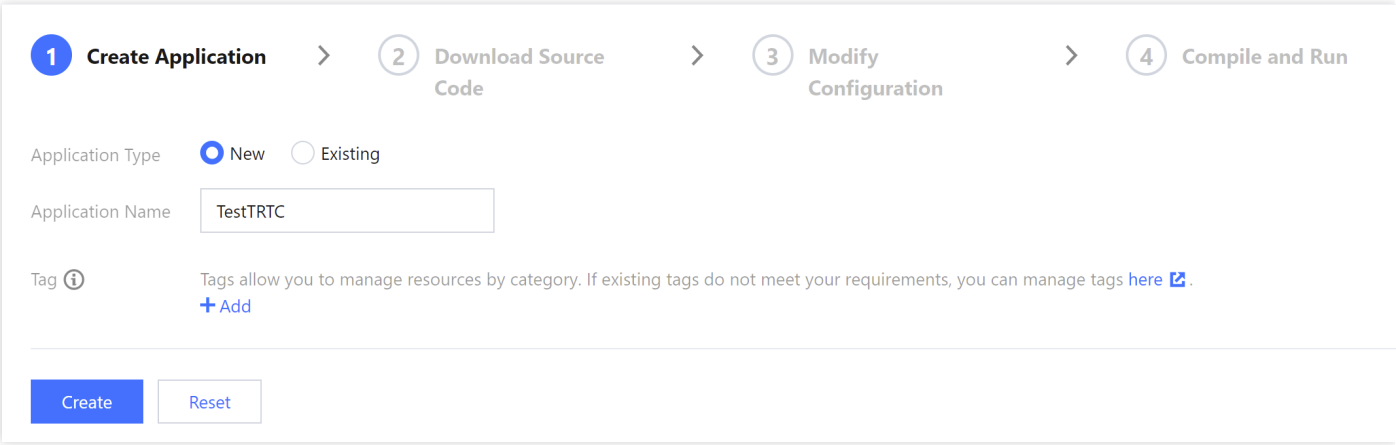
## 前提条件

您已 注册腾讯云 账号，并完成 实名认证。

## 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择**开发辅助**>快速跑通Demo。
2. 单击**新建应用**输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击**选择已有应用**。
3. 根据实际业务需求添加或编辑标签，单击**创建**。

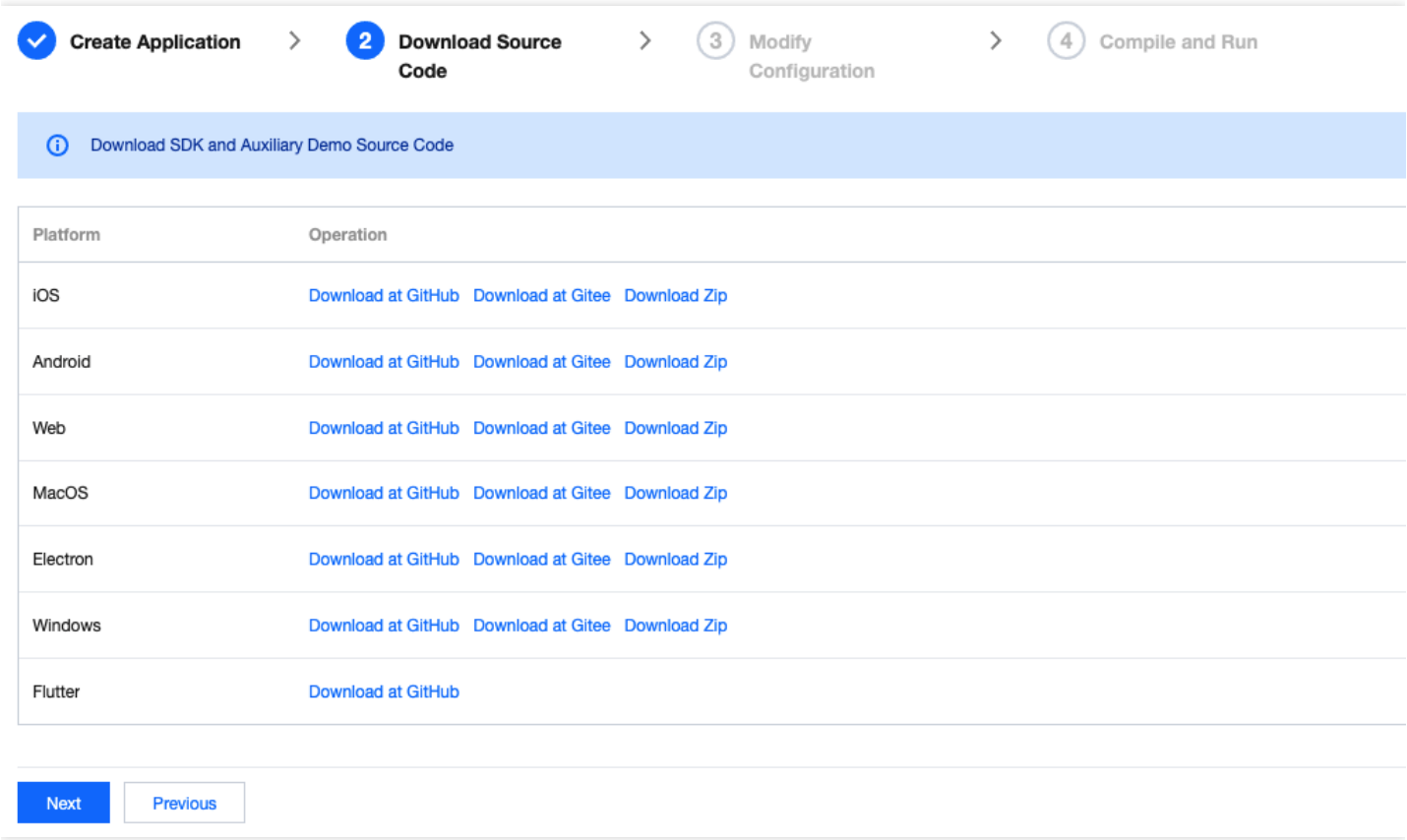



说明：

- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

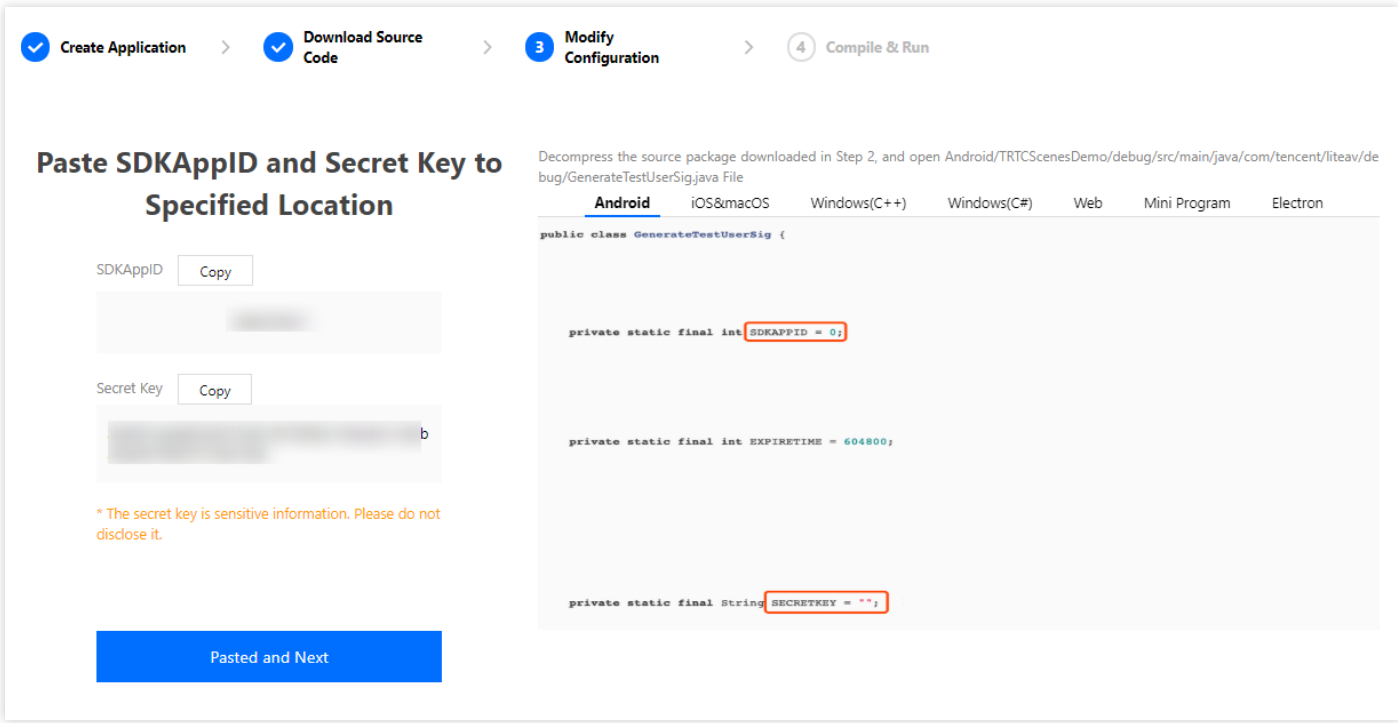
### 步骤2：下载 SDK 和 Demo 源码

1.根据实际业务需求下载 SDK 及配套的 Demo 源码。
2.下载完成后，单击**已下载，下一步**。




### 步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `Electron/js/GenerateTestUserSig.js` 文件。
3. 设置 `GenerateTestUserSig.js` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。




4. 粘贴完成后，单击**已复制粘贴，下一步**即创建成功。
5. 编译完成后，单击**回到控制台概览**即可。

注意：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

**文件目录说明：**

```
.
|---README.md README 文件，请详细阅读
|---main.electron.js Electron 主文件
|---public 存放静态文件
|---babel.config.js
|---package.json
|---vue.config.js vue-cli 工程文件
|---src 源代码目录
| |---app.vue
```

```
| |---common.css
| |---main.js
| |---components UI 组件目录
| | |---main-menu.vue
| | |---nav-bar.vue
| | |---show-screen-capture.vue
| |---common 工具函数、公共库等
| | |---live-room-service.js
| | |---log.js 日志工具
| | |---mtah5.js
| | |---routes.js
| | |---rand.js
| |---pages 页面目录
| | |---index.vue 主页
| | |---trtc 视频会议相关页面
| | | |---trtc-room.vue 视频会议房间页面
| | | |---trtc-index.vue 视频会议入口页
| | |---404.vue
| | |---live 直播页
| | | |---live-index.vue 直播入口页
| | | |---live-room-audience.vue 观众席页
| | | |---live-room-anchor.vue 主播间页
| |---debug 注意！在部署时，请将此文件夹内的签名逻辑移到服务器实现
| | |---lib-generate-test-usersig.min.js
| | |---gen-test-user-sig.js
```

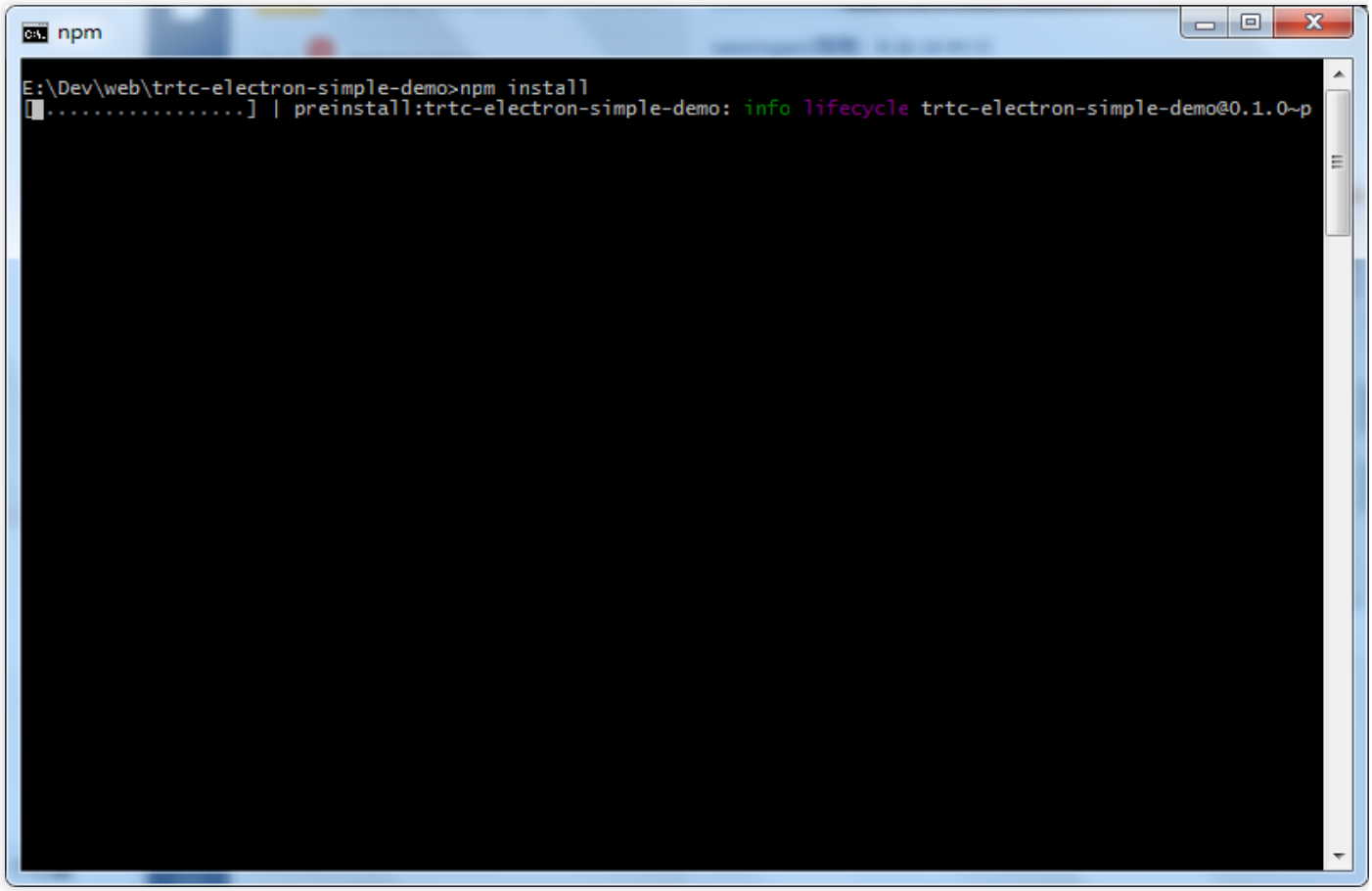
### 步骤4：编译运行

- Windows平台
- MacOS平台

1. 安装 Node 最新版本，建议选择 64bit 的 `.msi` 文件。Node 下载地址。

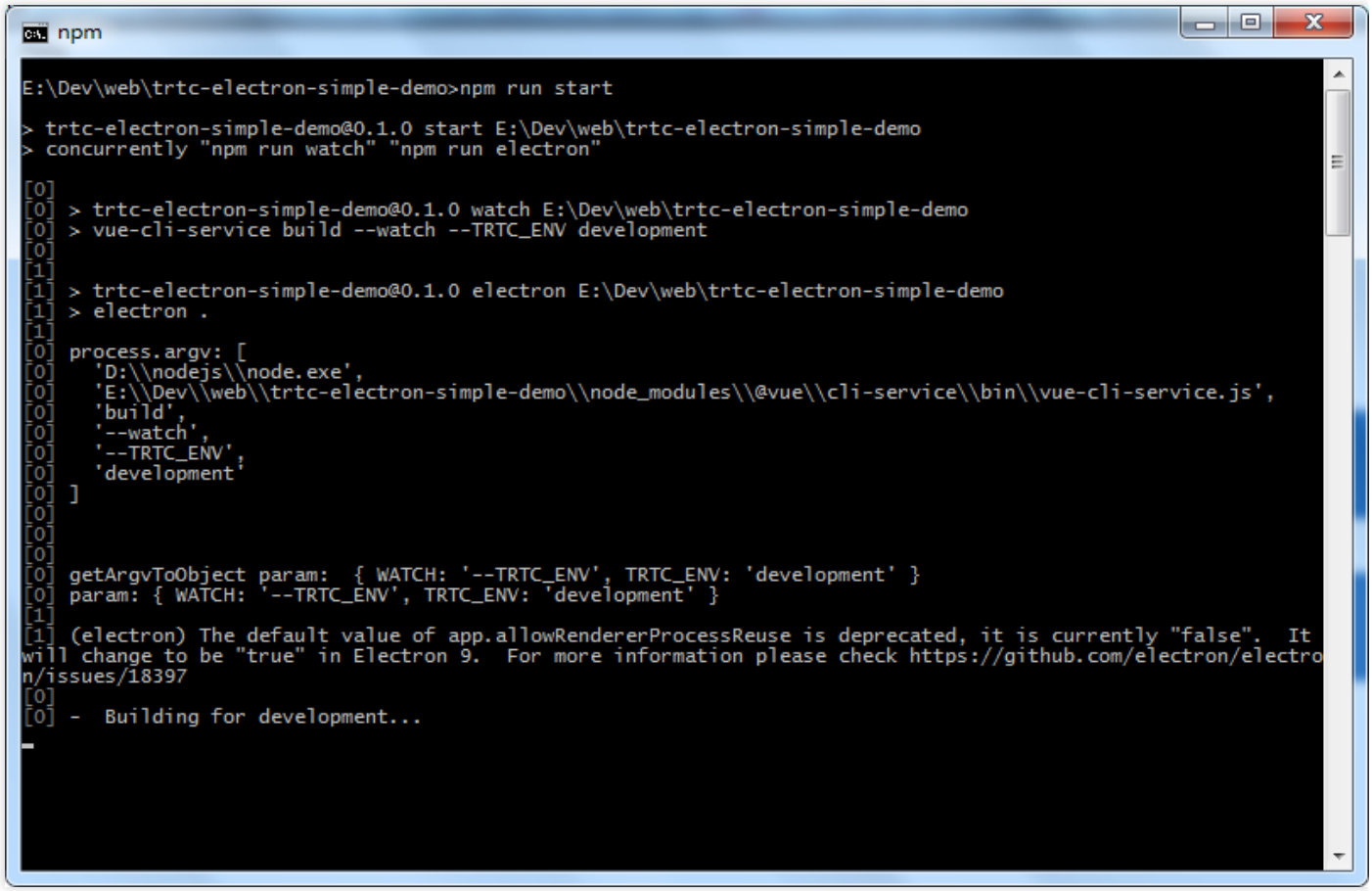
2. 按下 `win + r` 输入 cmd，用管理员权限启动命令行窗口，并将目录定位到 项目目录，并执行以下命令。

```
$ npm install
```




3. 待 npm 的依赖包都安装完成后，继续在命令行窗口执行以下命令，运行 Demo。

```
$ npm run start # 首次运行，稍等片刻后，窗口中才会出现 UI
```




**项目主要命令**

| 命令 | 说明 |
| --- | --- |
| npm run start | 以开发环境运行 Demo |
| npm run pack:mac | 打包 Mac 的 .dmg 安装文件 |
| npm run pack:win64 | 打包 Windows 64 位的 .exe 安装文件 |

# 常见问题

### 1. 查看密钥时只能获取公钥和私钥信息，要如何获取密钥？

TRTC SDK 6.6 版本（2019年08月）开始启用新的签名算法 HMAC-SHA256。在此之前已创建的应用，需要先升级签名算法才能获取新的加密密钥。如不升级，您也可以继续使用 老版本算法 ECDSA-SHA256。

**升级操作：**

1. 登录 实时音视频控制台。

2. 在左侧导航栏选择**应用管理**，单击目标应用所在行的**应用信息**。
3. 选择**快速上手**页签，单击**第二步 获取签发UserSig的密钥**区域的**点此升级**。

### 2. 两台设备同时运行 Demo，为什么看不到彼此的画面？

请确保两台设备在运行 Demo 时使用的是不同的 UserID，TRTC 不支持同一个 UserID （除非 SDKAppID 不同）在两个设备同时使用。

### 3. 防火墙有什么限制？

由于 SDK 使用 UDP 协议进行音视频传输，所以对 UDP 有拦截的办公网络下无法使用，如遇到类似问题，请参见应对公司防火墙限制。

> 说明：
> 更多相关问题，请参见 Electron 相关常见问题。

## 技术咨询

了解更多详情您可以联系我们。

## 参考文档

- SDK API 手册
- SDK 更新日志
- Simple Demo 源码
- API Example 源码
- Electron 常见问题

# Flutter

最近更新时间：2022-03-30 16:59:22

本文主要介绍如何快速运行腾讯云 TRTC Demo（Flutter）。

> 注意：
> 目前Flutter SDK仅支持Android和iOS

## 环境要求

- Flutter 2.0 及以上版本。
- **Android 端开发：**
  - Android Studio 3.5及以上版本。
  - App 要求 Android 4.1及以上版本设备。
- **iOS & macOS 端开发：**
  - Xcode 11.0及以上版本。
  - osx 系统版本要求 10.11 及以上版本
  - 请确保您的项目已设置有效的开发者签名。
- **Windows 开发：**
  - 操作系统：Windows 7 SP1 或更高的版本（基于 x86-64 的 64 位操作系统）。
  - 磁盘空间：除安装 IDE 和一些工具之外还应有至少 1.64 GB 的空间。
  - 安装 Visual Studio 2019。

## 前提条件

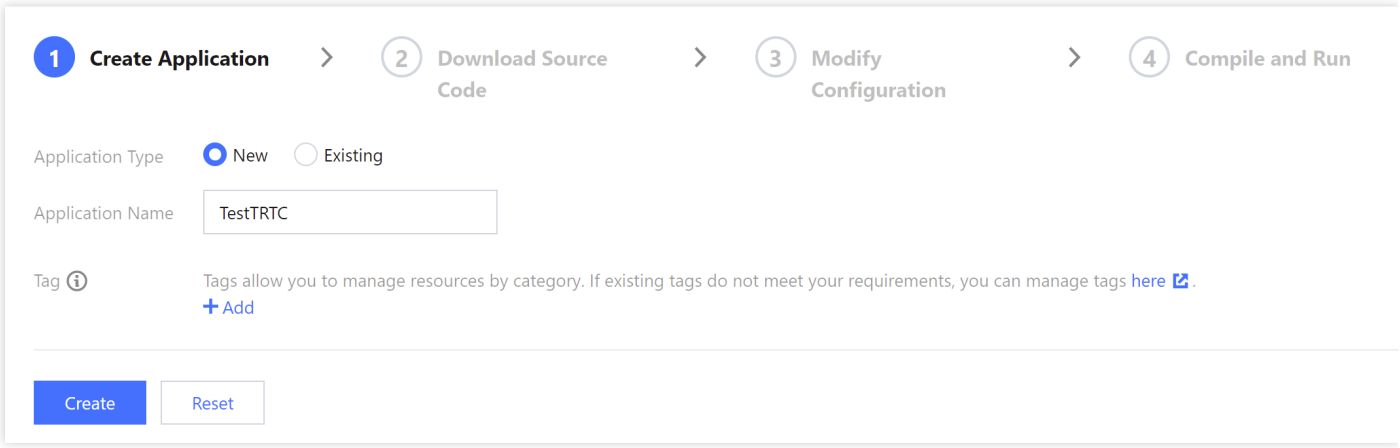
您已 注册腾讯云 账号，并完成实名认证。

## 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 单击【新建应用】输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击【选择已有应用】。

3. 根据实际业务需求添加或编辑标签，单击【创建】。



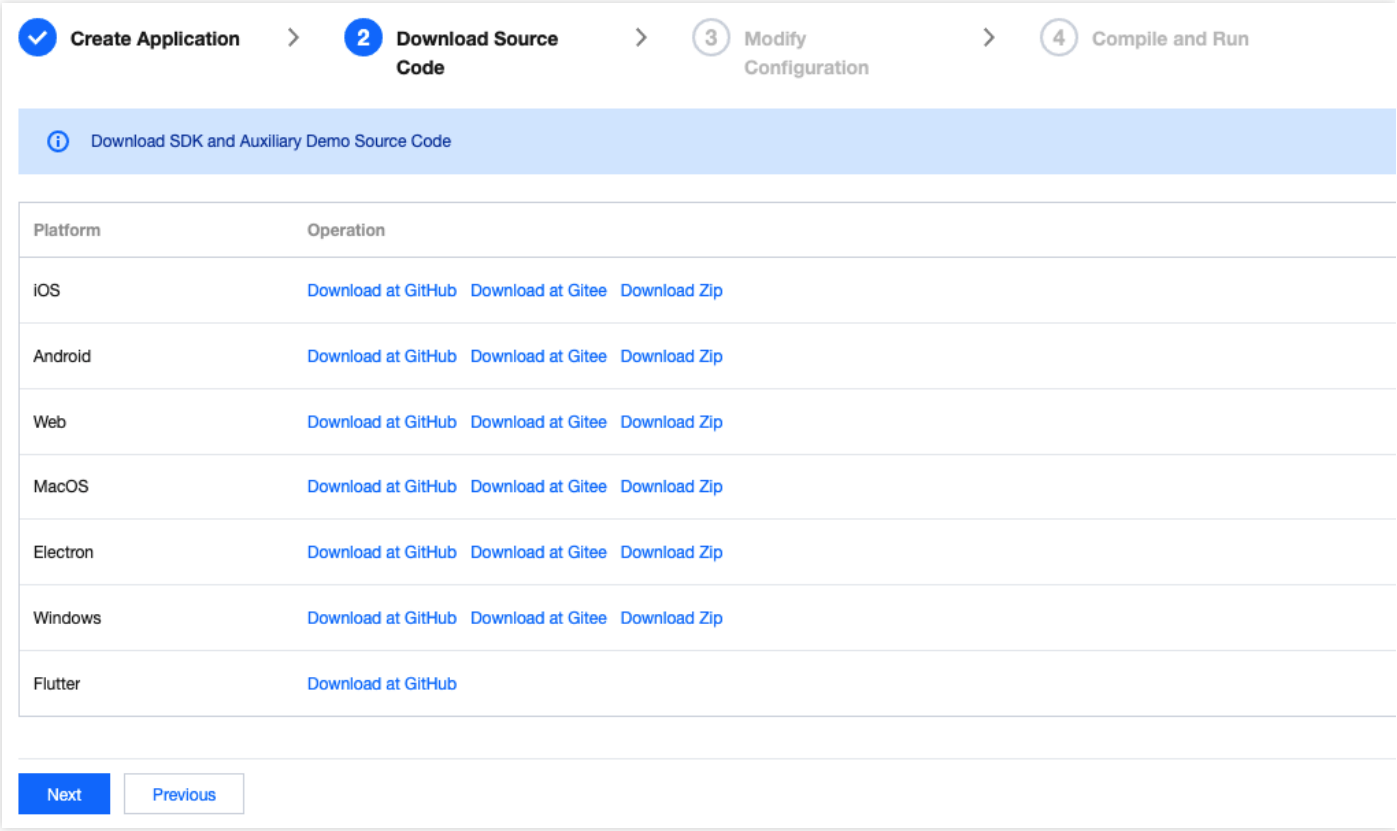

说明：

- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

### 步骤2：下载 SDK 和 Demo 源码

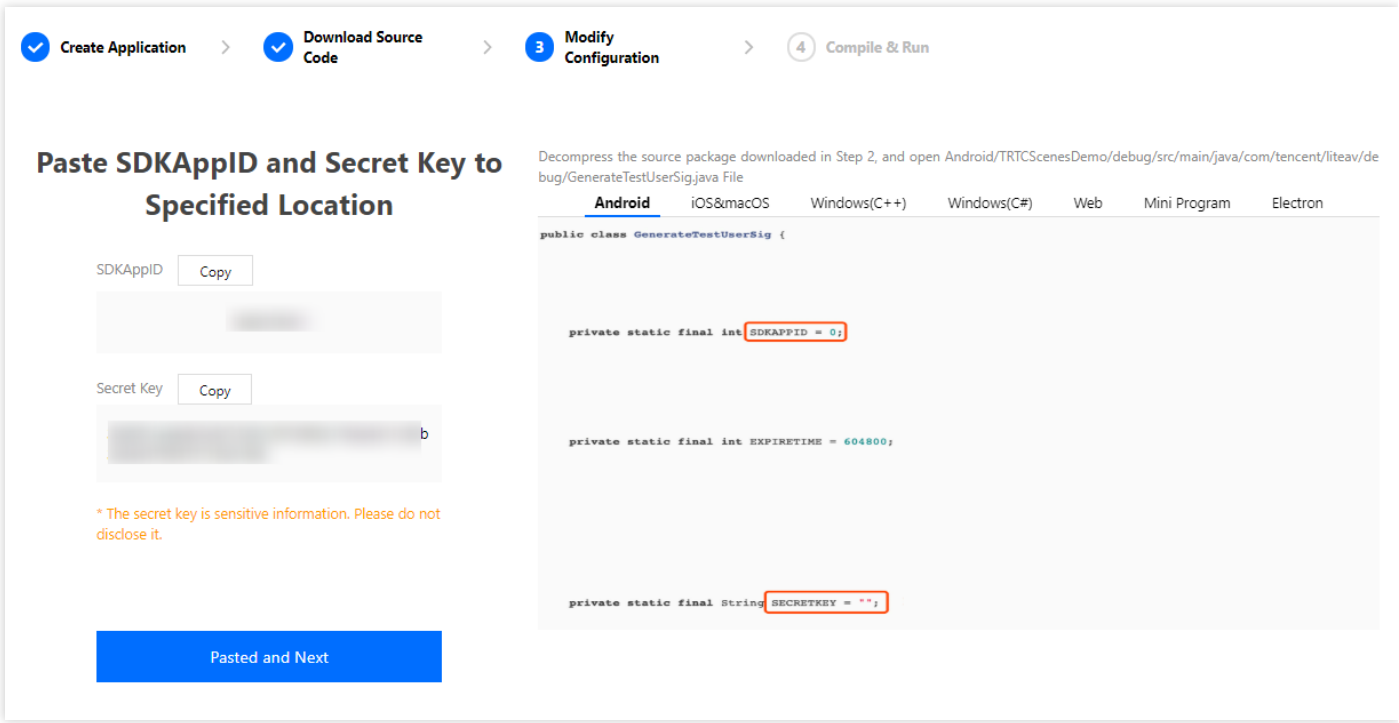
1. 根据实际业务需求下载 SDK 及配套的 Demo 源码。

2. 下载完成后，单击【已下载，下一步】。




### 步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `TRTC-Simple-Demo/example/lib/debug/GenerateTestUserSig.dart` 文件。
3. 设置 `GenerateTestUserSig.dart` 文件中的相关参数：
   - SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为 PLACEHOLDER，请设置为实际的密钥信息。




4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。
5. 编译完成后，单击【回到控制台概览】即可。

说明：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

### 步骤4：编译运行

1. 执行 `flutter pub get` 。
2. 编译运行调试：
   - Android 端
   - iOS 端
   - Windows 端
   - macOS 端

   i. 执行 `flutter run` 。

   ii. 使用 Android Studio（3.5及以上的版本）打开源码工程，单击【运行】即可。

# 常见问题

## 如何查看 TRTC 日志？

TRTC 的日志默认压缩加密，后缀为 `.xlog` 。地址如下：

- **iOS 端**：sandbox 的 `Documents/log` 。
- **Android 端**：
  - 6.7及之前的版本： `/sdcard/log/tencent/liteav` 。
  - 6.8之后的版本： `/sdcard/Android/data/包名/files/log/tencent/liteav/` 。

## iOS 无法显示视频（Android 没问题）？

请确认在您的 info.plist 中， `io.flutter.embedded_views_preview` 是否为 YES。

## Android Manifest merge failed 编译失败？

请打开 `/example/android/app/src/main/AndroidManifest.xml` 文件。

1. 将 `xmlns:tools="http://schemas.android.com/tools"` 加入到 manifest 中。
2. 将 `tools:replace="android:label"` 加入到 application 中。

```
android > app > src > main > AndroidManifest.xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mlp">
    <!-- io.flutter.app.FlutterApplication is an android.app.Application that
         calls FlutterMain.startInitialization(this); in its onCreate method.
         In most cases you can leave this as-is, but you if you want to provide
         additional functionality it is fine to subclass or reimplement
         FlutterApplication and put your custom class here. -->
    <application
        tools:replace="android:label"
        android:name="io.flutter.app.FlutterApplication"
        android:label="mlp"
        android:icon="@mipmap/ic_launcher">
```

说明：

更多常见问题，请参见 Flutter 相关问题。

# Unity

最近更新时间：2022-03-30 18:17:13

这个示例项目演示了如何在 Unity 中快速集成 TRTC SDK，实现在游戏中的音视频通话。

在这个示例项目中包含了以下功能：

- 加入通话和离开通话。
- 自定义视频渲染。
- 设备管理、音乐特效和人声特效。

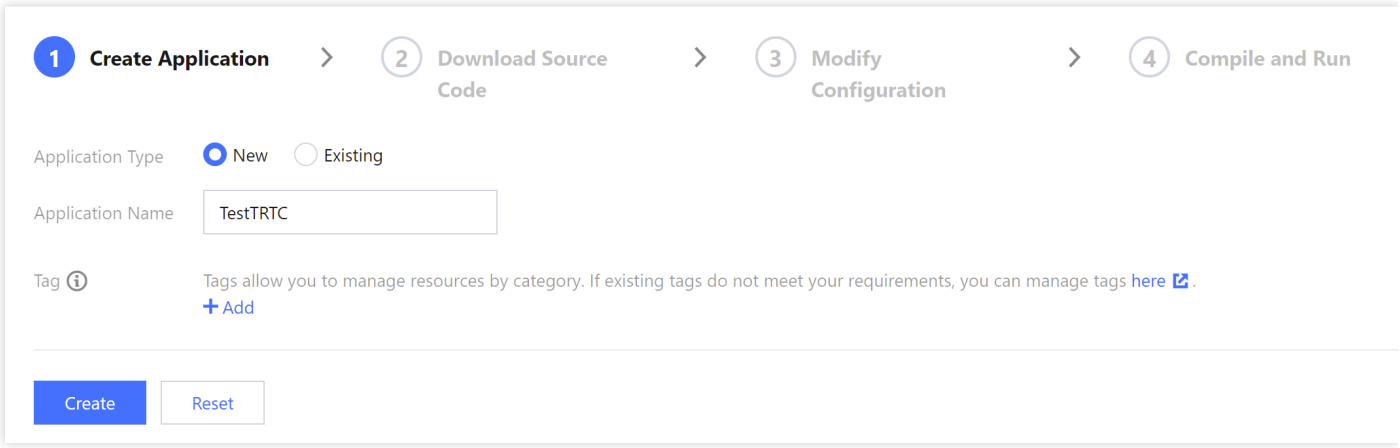
> 说明：
>
> - 具体 API 功能参数说明，请参见 Unity API 概览。
> - Unity 建议版本： 2020.2.1f1c1。
> - 目前支持 Android、iOS、Windows、Mac(Mac 还在内测中)平台。
> - 需要包含 `Android Build Support`、`iOS Build Support`、`Winodows Build Support` 和 `MacOs Build Support` 模块。
> - 其中 iOS 端开发还需要：
>   - Xcode 11.0及以上版本。
>   - 请确保您的项目已设置有效的开发者签名。

## 运行示例程序

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。

2. 输入应用名称，例如 `TestTRTC`，单击【创建应用】。




### 步骤2：下载 SDK 与源码

1. 根据您的实际业务需求，下载 SDK 及配套的 Demo 源码。

2. 下载完成后，单击【已下载，下一步】。（可直接用 Unity 打开本项目；如果想直接用 SDK 文件，也可把 SDK 包中的 `TRTCUnitySDK/Assets/TRTCSDK/SDK` 文件夹拷贝到您项目中的 Assets 目录下。）

3. 找到并打开 `Assets/TRTCSDK/Demo/Tools/GenerateTestUserSig.cs` 文件。

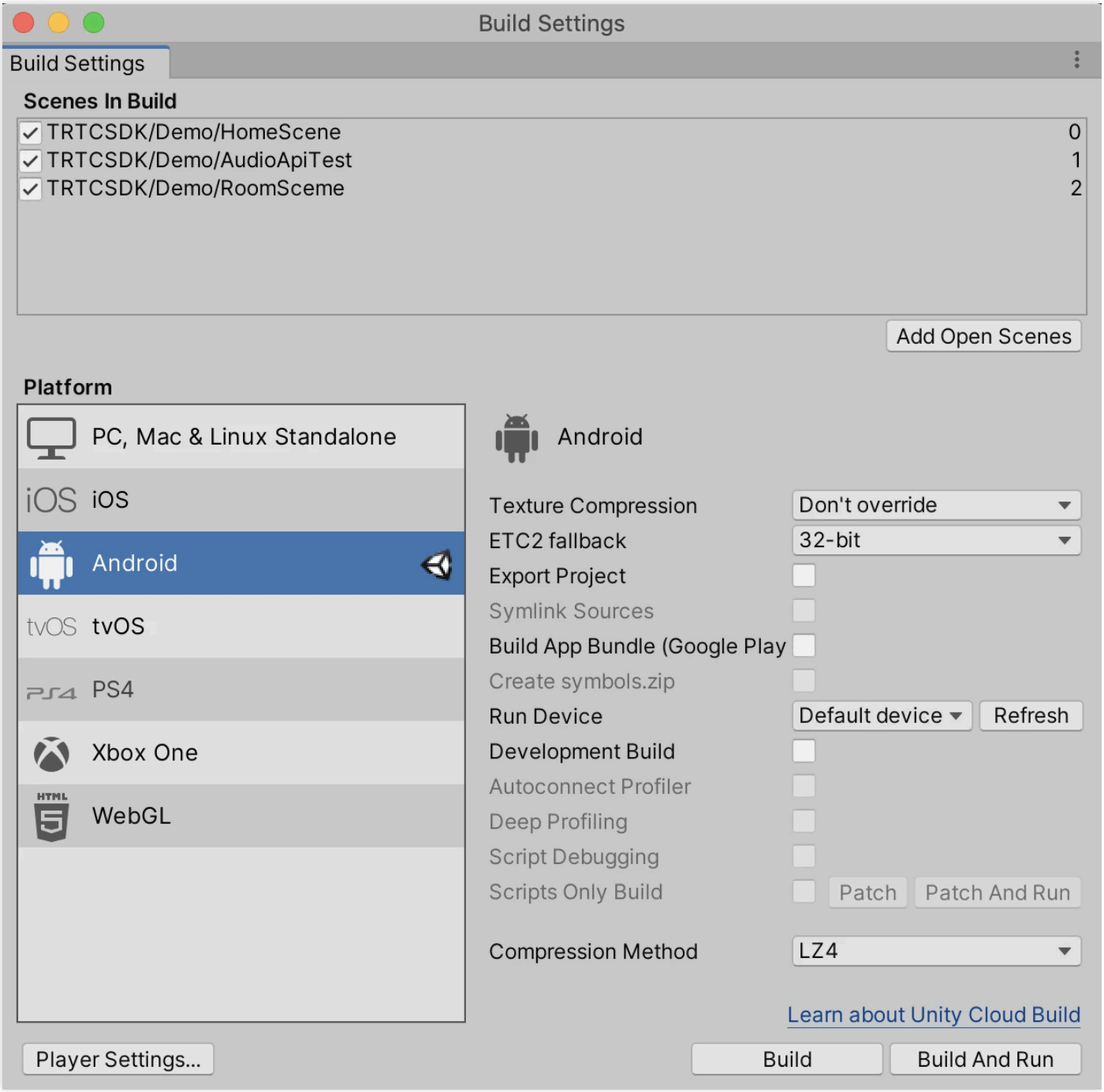
4. 设置 `GenerateTestUserSig.cs` 文件中的相关参数：

   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。

### 步骤3：编译运行

- Android 平台
- iOS 平台
- Windows 平台
- macOS 平台

1. 配置 Unity Editor，单击【File】>【Build Setting】，切换至 Android。




2. 连接 Android 真机，单击【 Build And Run】， Demo 就能跑起来。
3. 接口测试，需要先点击调用 enterRoom ， 然后自行测试其他相关，数据展示窗口显示点击调用成功，另外一个窗口显示回调信息。

# Demo示例

Demo 里面包含了已上线的大部分 API，可以测试和作为调用参考，API 文档参见 SDK API（Unity）。

说明：
UI 可能会有部分调整更新，请以最新版为准。

# 目录结构

```
├──Assets
├─── Editor // Unity 编辑器脚本
|  ├─── BuildScript.cs // Unity 编辑器build菜单
|  ├─── IosPostProcess.cs // Unity 编辑器构建ios应用脚本
├─── Plugins
|  ├─── Android
|  |  ├─── AndroidManifest.xml //Android应用配置文件
├─── StreamingAssets // Unity Demo 音视频流文件
├─── TRTCSDK
├─── Demo // Unity 示例 Demo
├─── SDK // TRTC Unity SDK
├─── Implement // TRTC Unity SDK 实现
├─── Include // TRTC Unity SDK 头文件
└─── Plugins // TRTC Unity SDK 不同平台底层实现
```

# React Native

最近更新时间：2022-04-02 11:49:37

本文主要介绍如何快速运行腾讯云 TRTC Demo（React Native）。

## 环境要求

- ReactNative 0.63 及以上版本
- Node & Watchman，node版本需在 v12 以上
- **Android 端开发：**
  - Android Studio 3.5及以上版本
  - App 要求 Android 4.1及以上版本设备
- **iOS & macOS 端开发：**
  - Xcode 11.0及以上版本
  - osx 系统版本要求 10.11 及以上版本
  - 请确保您的项目已设置有效的开发者签名
- 环境安装请参见 官方文档

## 前提条件

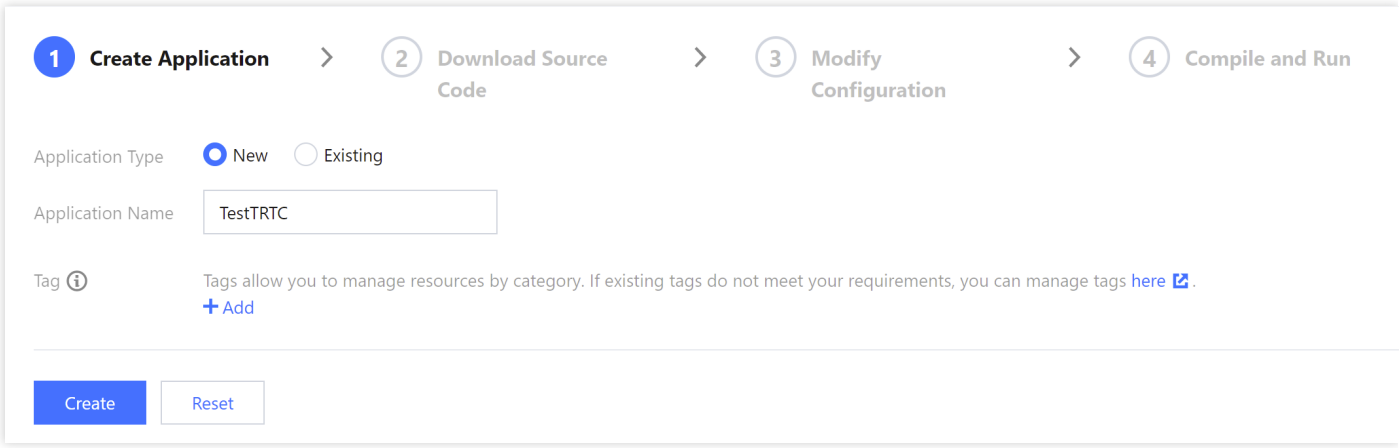
您已 注册腾讯云 账号，并完成实名认证。

## 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择 开发辅助>快速跑通Demo。
2. 单击 **新建应用** 输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击 **选择已有应用**。

3. 根据实际业务需求添加或编辑标签，单击 **创建**。




说明：

- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

### 步骤2：下载 SDK 和 Demo 源码

1. 根据实际业务需求下载 SDK 及配套的 Demo 源码。
2. 下载完成后，单击 **已下载，下一步**。

注意

控制台暂时无法下载 ReactNative Demo，**请直接通过上方链接下载 Demo 源码**。

### 步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `/debug/config.js` 文件。
3. 设置 `SDKAPPID` 和 `SECRETKEY` 参数：
   - SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID。

- SECRETKEY：默认为 PLACEHOLDER ， 请设置为实际的密钥信息。

4. 粘贴完成后，单击 **已复制粘贴，下一步** 即创建成功。

5. 编译完成后，单击 **回到控制台概览** 即可。

> 说明：
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
> - 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

### 步骤4：权限配置

需要配置 App 权限才能运行。

- Android 端
- iOS 端

1. 在 `AndroidManifest.xml` 中配置 App 的权限，TRTC SDK 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-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

> 注意

请勿设置 `android:hardwareAccelerated="false"`，关闭硬件加速之后，会导致对方的视频流无法渲染。

2. Android 端音视频权限需要手动申请。

```
if (Platform.OS === 'android') {
await PermissionsAndroid.requestMultiple([
PermissionsAndroid.PERMISSIONS.RECORD_AUDIO, //音频需要
PermissionsAndroid.PERMISSIONS.CAMERA, // 视频需要
]);
}
```

### 步骤5：编译运行

启动 Metro，在您的 React Native 项目目录下运行 `npx react-native start` 。

- Android 端
- iOS 端

新开窗口，启动开发调试：

```
npx react-native run-android
```

# Unreal Engine

最近更新时间：2022-03-30 18:08:51

本文主要介绍如何快速运行腾讯云 TRTC Demo（Unreal Engine）。

> 注意：
> 目前 支持Windows、MacOs、ios、Android。

## 环境要求

- 建议Unreal Engine 4.27.1 及以上版本。
- **Android 端开发：**
  - Android Studio版本4.0及以上版本。
  - Visual Studio 2017 15.6版或更高。
  - 只支持真机调试
- **iOS & macOS 端开发：**
  - Xcode 11.0及以上版本。
  - osx 系统版本要求 10.11 及以上版本
  - 请确保您的项目已设置有效的开发者签名。
- **Windows 开发：**
  - 操作系统：Windows 7 SP1 或更高的版本（基于 x86-64 的 64 位操作系统）。
  - 磁盘空间：除安装 IDE 和一些工具之外还应有至少 1.64 GB 的空间。
  - 安装 Visual Studio 2019。

## 前提条件

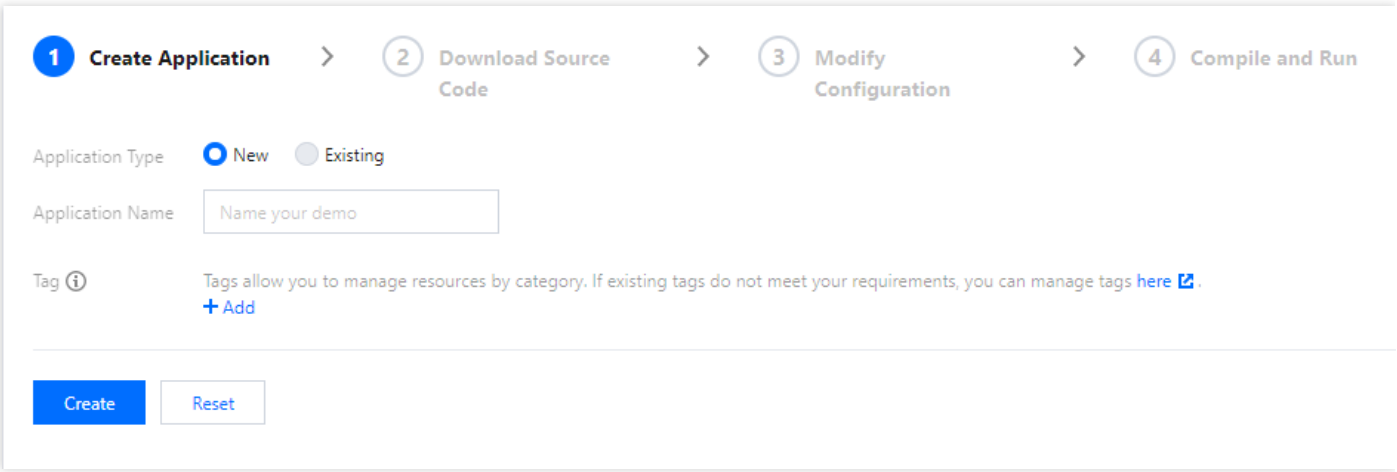
您已 注册腾讯云 账号，并完成实名认证。

## 操作步骤

### 步骤1：创建新的应用

1. 登录实时音视频控制台，选择【开发辅助】>【快速跑通Demo】。
2. 单击【新建应用】输入应用名称，例如 `TestTRTC` ；若您已创建应用可单击【选择已有应用】。

3. 根据实际业务需求添加或编辑标签，单击【创建】。




说明：

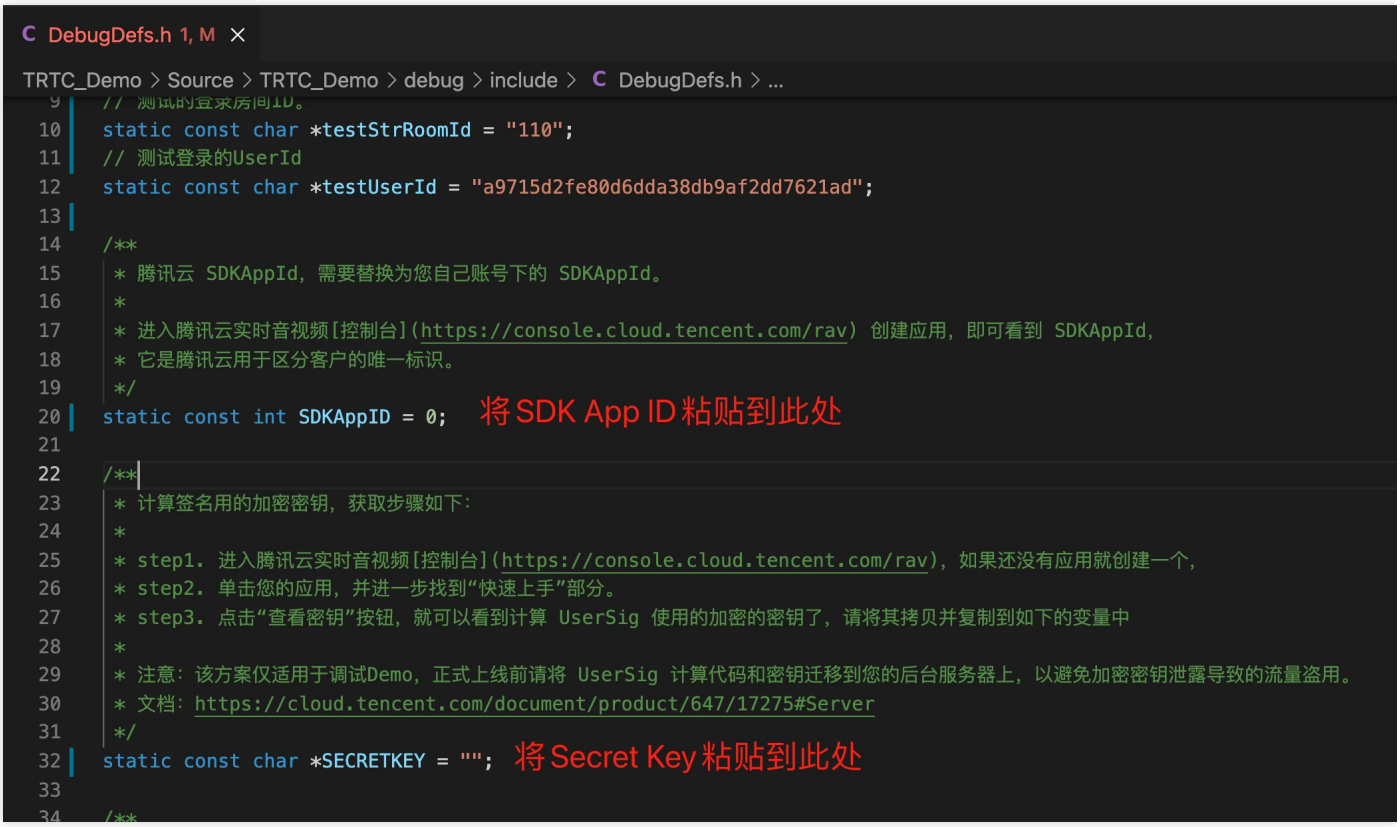
- 应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。
- 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

### 步骤2：下载 SDK 和 Demo 源码

1. 根据实际业务需求下载 SDK 及配套的 Demo 源码（有疑问可在此处提issue单）。
2. 下载完成后，单击【已下载，下一步】。

### 步骤3：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `/TRTC_Demo/Source/debug/include/DebugDefs.h` 文件。
3. 设置 `DebugDefs.h` 文件中的相关参数：
   - SDKAPPID：默认为 0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为 ""，请设置为实际的密钥信息。




4. 粘贴完成后，单击【已复制粘贴，下一步】即创建成功。
5. 编译完成后，单击【回到控制台概览】即可。

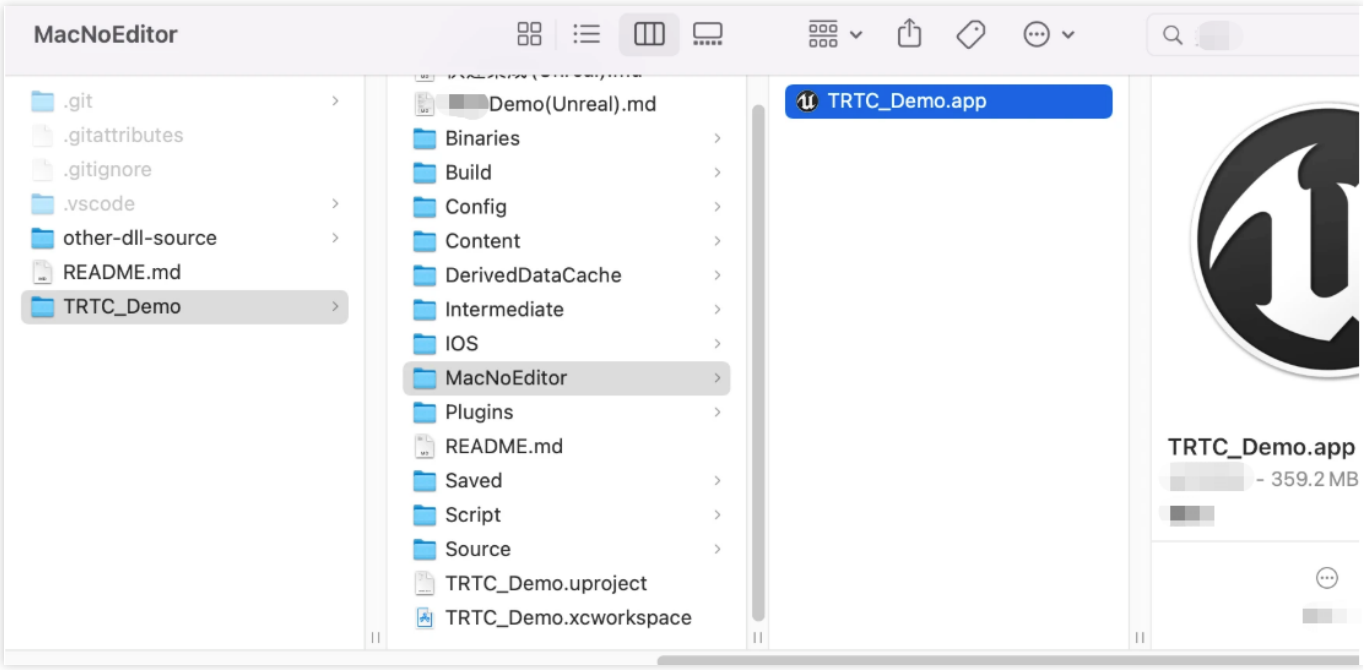
说明：

- 本文提到的生成 UserSig 的方案是在客户端代码中配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。
- 正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

### 步骤4：编译打包运行

1. 双击打开 `/TRTC_Demo/TRTC_Demo.uproject` 。
2. 编译运行调试：
   - macOS 端
   - Windows 端
   - iOS 端
   - Android 端

i. File -> Package Project -> Mac
ii. 配置权限。右击上一步编译出的 xxx.app 文件 - 选择 "显示包内容"




iii. 进入 "Contents->Info.plist"
iv. 选择 "Information Property List" 然后添加以下两个权限:

```
<key>NSCameraUsageDescription</key>
<string>授权摄像头权限才能正常视频通话</string>
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

v. 如果你现在在UE4的editor运行的话，需要找到 **UE4Editor.app** 文件并且按照上面步骤添加权限。

# Demo运行

Demo 里面提供了一对一视频通话的实现，可以测试和作为调用参考，API 文档参见 C++ 全平台 API。

> 说明：
> UI 可能会有部分调整更新，请以最新版为准。


# TRTC全平台 C++ API文档

中文文档

英文文档

# 常见问题

### 如何查看 TRTC 日志？

TRTC 的日志默认压缩加密，后缀为 `.xlog` 。地址如下：

- **iOS 端**：sandbox 的 `Documents/log` 。
- **Android 端**：
  - 6.7及之前的版本： `/sdcard/log/tencent/liteav` 。
  - 6.8之后的版本： `/sdcard/Android/data/包名/files/log/tencent/liteav/` 。

### macos UE4 editor闪退

请确认在您的**UE4Editor.app** info.plist 中设置了音视频权限

```
<key>NSCameraUsageDescription</key>
<string>授权摄像头权限才能正常视频通话</string>
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

### 安卓“Attempt to construct staged filesystem reference from absolute path""”报错

关闭UE4项目，打开cmd

```
adb shell
cd sdcard
ls (you should see the UE4Game directory listed)
rm -r UE4Game
重新编译项目
```

# 新手常见问题

最近更新时间：2022-01-26 16:28:19

### 什么是 UserSig？

UserSig 是腾讯云设计的一种安全保护签名，目的是为了阻止恶意攻击者盗用您的云服务使用权。
目前，腾讯云的实时音视频（TRTC）、即时通信（IM）以及移动直播（MLVB）等服务都采用了该套安全保护机制。要使用这些服务，您都需要在相应 SDK 的初始化或登录函数中提供 SDKAppID，UserID 和 UserSig 三个关键信息。
其中 SDKAppID 用于标识您的应用，UserID 用于标识您的用户，而 UserSig 则是基于前两者计算出的安全签名，它由 **HMAC SHA256** 加密算法计算得出。只要攻击者不能伪造 UserSig，就无法盗用您的云服务流量。
UserSig 的计算原理如下图所示，其本质就是对 SDKAppID、UserID、ExpireTime 等关键信息进行了一次哈希加密：

```
//UserSig 计算公式，其中 secretkey 为计算 usersig 用的加密密钥
usersig = hmacsha256(secretkey, (userid + sdkappid + currtime + expire +
base64(userid + sdkappid + currtime + expire)))
```

说明：

- `currtime` 为当前系统的时间，`expire` 为签名过期的时间。
- 如需了解 UserSig 具体计算获取方法，请参见 UserSig 详情说明。

### 实时音视频最多可以同时创建多少个房间？

支持同时并发存在4294967294个房间，累计房间数量无限制。

### 实时音视频延时大约多少？

全球端到端平均延时小于300ms。

### 实时音视频接入 PC 端是否支持屏幕分享功能？

支持，您可以参考如下文档：

- 屏幕分享（Windows）
- 屏幕分享（Mac）
- 屏幕分享（Web）

屏幕分享接口详情请参见 Windows（C++）API 。另外，您也可以使用 Electron 接口。

### TRTC 支持哪些平台？

支持的平台包括 iOS、Android、Windows(C++)、Windows(C#)、Mac、Web、Electron，更多详情请参见 平台支持。

### 实时音视频最多可以支持多少个人同时通话？

- 通话模式下，单个房间最多支持300人同时在线，最多支持50人同时开启摄像头或麦克风。
- 直播模式下，单个房间支持10万人以观众身份在线观看，最多支持50人以主播身份开启摄像头或麦克风。

### TRTC 怎么实现直播场景类应用？

TRTC 专门针对在线直播场景推出了10万人低延时互动直播解决方案，能保证主播与连麦主播的最低延时到200ms，普通观众的延时在1s以内，并且超强的抗弱网能力适应移动端复杂的网络环境。
具体操作指引请参考 跑通直播模式。

### TRTC 直播支持什么角色？有什么区别？

直播场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom）支持 TRTCRoleAnchor（主播）和 TRTCRoleAudience（观众）两种角色，区别是主播角色可以同时上行、下行音视频数据，观众角色只支持下行播放其他人的数据。您可以通过调用 switchRole() 进行角色切换。

### TRTC 房间支不支持踢人、禁止发言、静音？

支持。

- 如果是简单的信令操作，可以使用 TRTC 的自定义信令接口 sendCustomCmdMsg，开发者自己定义相应的控制信令，收到控制信令的通话方执行对应操作即可。例如，踢人就是定义一个踢人的信令，收到此信令的用户就自行退出房间。
- 如果是需要实现更完善的操作逻辑，建议开发者通过 即时通信 IM 来实现相关逻辑，将 TRTC 的房间与 IM 群组进行映射，在 IM 群组中收发自定义消息来实现相应的操作。

### TRTC 音视频流是否支持通过 CDN 拉流观看？

支持，详情请参见 实现 CDN 直播观看。

### 在 iOS 端是否支持 Swift 集成？

支持，直接按照支持集成三方库的流程集成 SDK 即可，还可以参考 跑通Demo(iOS&Mac)。

### Web 端 SDK 的支持哪些浏览器？

目前主要在桌面版 Chrome 浏览器、桌面版 Safari 浏览器以及移动版的 Safari 浏览器上有较为完整的支持，其他平台（例如 Android 平台的浏览器）支持情况均比较差，具体详情请参见 支持的平台。
您可以在浏览器打开 WebRTC 能力测试 测试是否完整的支持 WebRTC 的功能。

### Web 端 SDK 日志中报错 NotFoundError、NotAllowedError、NotReadableError、OverConstrainedError 以及 AbortError 分别是什么意思？

| 错误名 | 描述 | 处理建议 |
| --- | --- | --- |
| NotFoundError | 找不到满足请求参数的媒体类型（包括音频、视频、屏幕分享）。<br>例如：PC 没有摄像头，但是请求浏览器获取视频流，则会报此错误。 | 建议在通话开始前引导用户检查通话所需的摄像头或麦克风等设备，若没有摄像头且需要进行语音通话，可在 TRTC.createStream({ audio: true, video: false }) 指明仅采集麦克风。 |
| NotAllowedError | 用户拒绝了当前的浏览器实例的访问音频、视频、屏幕分享请求。 | 提示用户不授权摄像头/麦克风访问将无法进行音视频通话。 |
| NotReadableError | 用户已授权使用相应的设备，但由于操作系统上某个硬件、浏览器或者网页层面发生的错误导致设备无法被访问。 | 根据浏览器的报错信息处理，并提示用户“暂时无法访问摄像头/麦克风，请确保当前没有其他应用请求访问摄像头/麦克风，并重试”。 |
| OverConstrainedError | cameraId/microphoneId 参数的值无效。 | 请确保 cameraId/microphoneId 传值正确且有效。 |
| AbortError | 由于某些未知原因导致设备无法被使用。 | - |

更多详情请参考 initialize。

### 怎么确认 Web 端 SDK 是否能正常获取到设备（摄像头/麦克风）列表？

1. 检查浏览器是否能够正常使用设备：
   直接在页面打开控制台，输入 `navigator.mediaDevices.enumerateDevices()` 确认能否获取到设备列表。
   - 如果正常获取到设备会返回一个 Promise，里面会有 MediaDeviceInfo 对象数组，数组里的每个对象对应一个可用的媒体设备。
   - 如果枚举失败，Promise 将返回 rejected，说明浏览器都没有识别到设备，需检查浏览器或设备。
2. 如果能获取设备列表，则输入 `navigator.mediaDevices.getUserMedia({ audio: true, video: true })` 确认能否正常返回 MediaStream 对象，不能正常返回说明浏览器没有获取到数据，需检查浏览器的配置。

### 直播、互动直播、实时音视频以及旁路直播有什么区别和关系？

- **直播**（关键词：一对多，RTMP/HLS/HTTP-FLV，CDN）
  直播分为推流端、播放端以及直播云服务，云服务使用 CDN 进行直播流的分发。推流使用的是通用标准的协议 RTMP，经过 CDN 分发后，播放时一般可以选择 RTMP、HTTP-FLV 或 HLS（H5 支持）等方式进行观看。

- **互动直播**（关键词：连麦、PK）
  互动直播是一种业务形式，指主播与观众之间进行互动连麦，主播与主播之间进行互动PK的一种直播类型。
- **实时音视频**（关键词：多人互动，UDP 私有协议，低延时）
  实时音视频（Tencent Real-Time Communication，TRTC）主要应用场景是音视频互动和低延时直播，使用基于 UDP 的私有协议，其延迟可低至100ms，典型的场景就是 QQ 电话、腾讯会议、大班课等。 腾讯云实时音视频（TRTC）覆盖全平台，除了 iOS/Android/Windows 之外，还支持 WebRTC 互通，并且支持通过云端混流的方式将画面旁路直播到 CDN。
- **旁路直播**（关键词：云端混流，RTC 旁路转推，CDN）
  旁路直播是一种技术，指的是将低延时连麦房间里的多路推流画面复制出来，在云端将画面混合成一路，并将混流后的画面推流给直播 CDN 进行分发播放。

### TRTC 如何查看通话时长和使用量？

可在实时音视频控制台的【用量统计】页面查看。

### TRTC 出现卡顿怎么排查？

可以通过对应的 RoomID、UserID 在实时音视频控制台的【监控仪表盘】页面查看通话质量：

- 通过接受端视角查看发送端和接收端用户情况。
- 查看发送端和接收端是否丢包率比较高，如果丢包率过高一般是网络状况不稳定导致卡顿。
- 查看帧率和 CPU 占用率，帧率比较低和 CPU 使用率过高都会导致卡顿现象。

### TRTC 出现画质不佳，模糊、马赛克等现象怎么排查？

- 清晰度主要和码率有关，检查 SDK 码率是否配置的比较低，如果高分辨率低码率容易产生马赛克现象。
- TRTC 会通过云端 QOS 流控策略，根据网络状况动态调整码率、分辨率，网络比较差时容易降低码率导致清晰度下降。
- 检查进房时使用的 VideoCall 模式还是 Live 模式，针对通话场景 VideoCall 模式主打低延时和保流畅，所以在弱网情况下会更容易牺牲画质确保流畅，对画质更加看重的场景建议使用 Live 模式。

### 如何查询 SDK 最新版本号？

- 若您使用自动加载的方法，`latest.release` 为匹配最新版并进行自动加载，不需要对版本号进行修改。具体集成方法请参见 一分钟集成 SDK。
- 当前 SDK 最新版本号可通过发布日志查看，具体请参见：
  - iOS & Android 端，请参见 发布日志（App）。
  - Web 端，请参见 发布日志（Web）。
  - Electron 端，请参见 发布日志（Electron）。