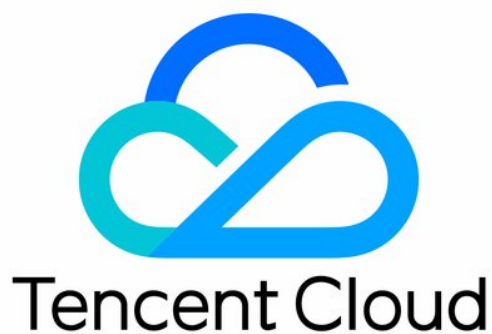


# **Tencent Real-Time Communication Getting Started Product Documentation**



## Copyright Notice

©2013-2019 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Getting Started

### SDK Quick Integration

iOS

Android

macOS

Windows

Web

Electron

Flutter

Qt Creator

Unity

React Native

Unreal Engine

### Demo Quick Start

iOS and macOS

Android

Windows

Web

Electron

Flutter

Unity

React Native

Unreal Engine

### FAQs for Beginners

# Getting Started

## SDK Quick Integration

### iOS

Last updated : 2022-04-02 16:27:02

This document describes how to quickly integrate the TRTC SDK for iOS into your project.

## Environment Requirements

- Xcode 9.0 or above
- iPhone or iPad with iOS 9.0 or above
- A valid developer signature for your project

## Integrating the TRTC SDK

You can use CocoaPods to automatically load the SDK or download and import it manually into your project.

### CocoaPods

#### 1. Install CocoaPods

Enter the following command in a terminal window (you need to install Ruby on your Mac first):

```
sudo gem install cocoapods
```

#### 2. Create a Podfile

Go to the directory of your project and enter the following command to create a Podfile in the directory.

```
pod init
```

#### 3. Edit the Podfile

```
platform :ios, '8.0'
```

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

You can use an official CocoaPods source, but the download may be slow:

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

target 'App' do
  pod 'TXLiteAVSDK_TRTC'
end
```

#### 4. Update the local repository and install the SDK

Enter the following command in a terminal window to update the local repository and install the SDK:

```
pod install
```

Or, run the following command to update the local repository:

```
pod update
```

An XCWORKSPACE project file integrated with the TRTC SDK will be generated. Double-click to open it.

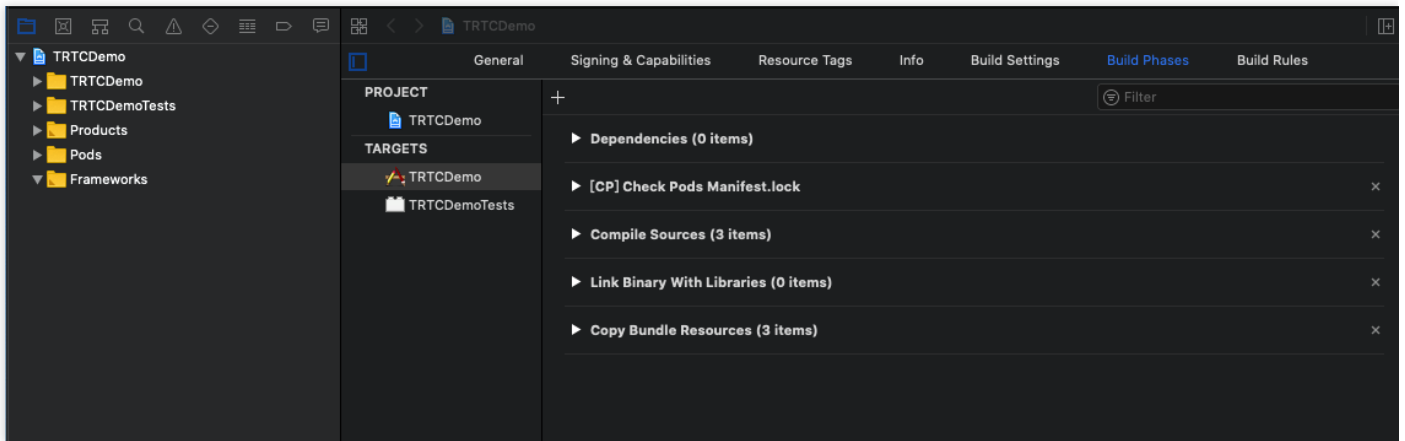
Note :

You need to manually add the dependent library **Accelerate.framework**.

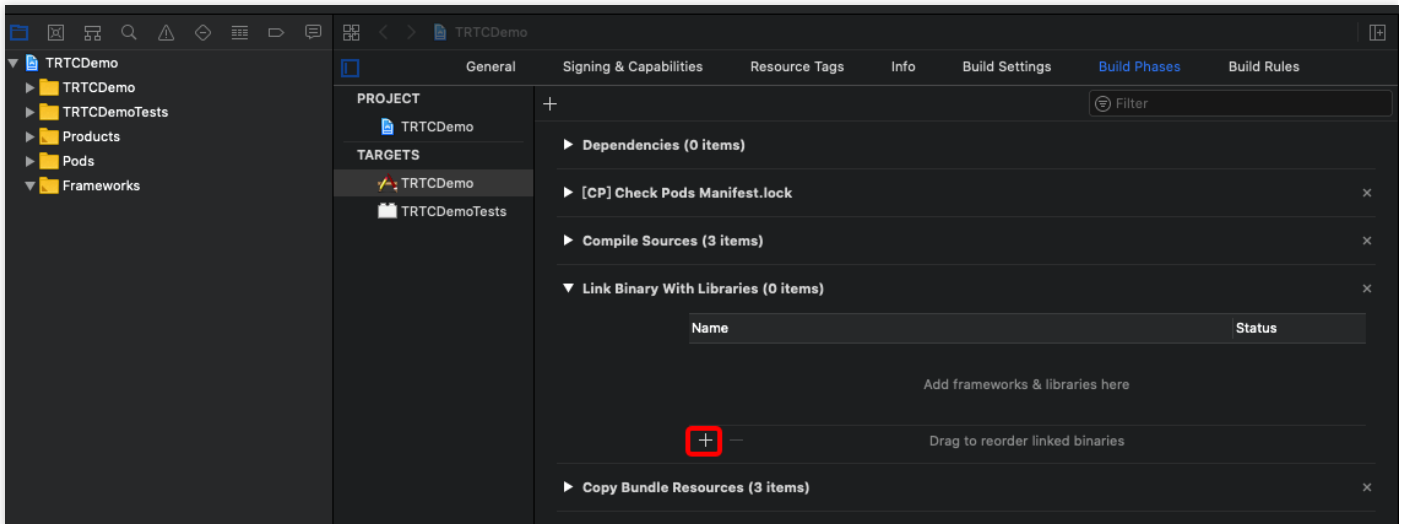
### Manual integration

1. Download the [TRTC SDK](#) and decompress the downloaded file.

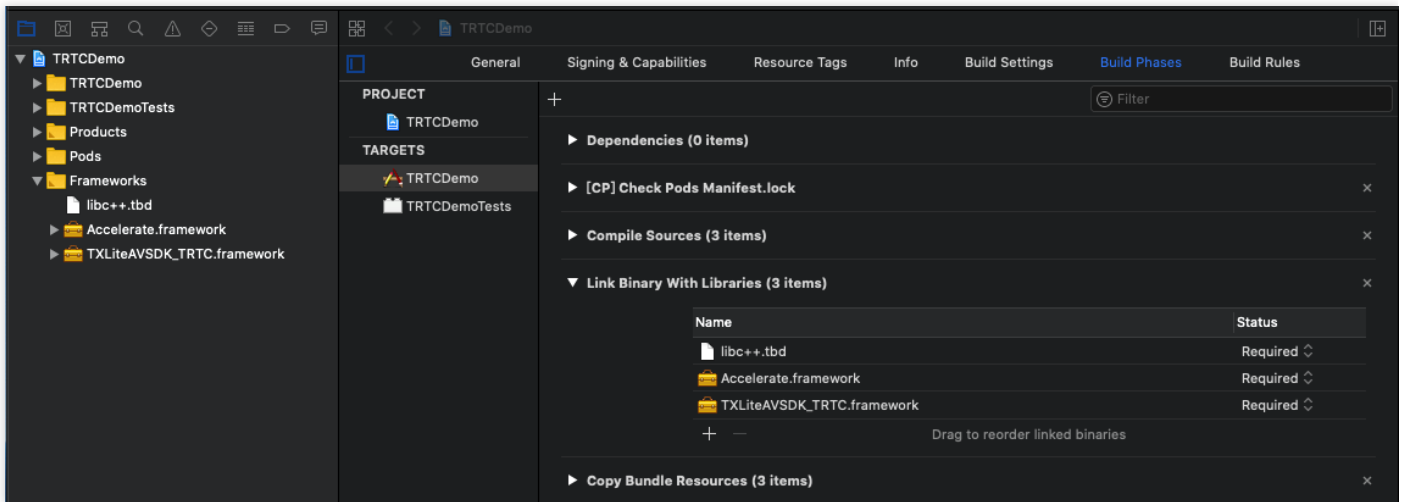
2. Open your Xcode project, select the target you want to run, and click **Build Phases**.



3. Expand **Link Binary With Libraries** and click the + icon at the bottom to add dependent libraries.

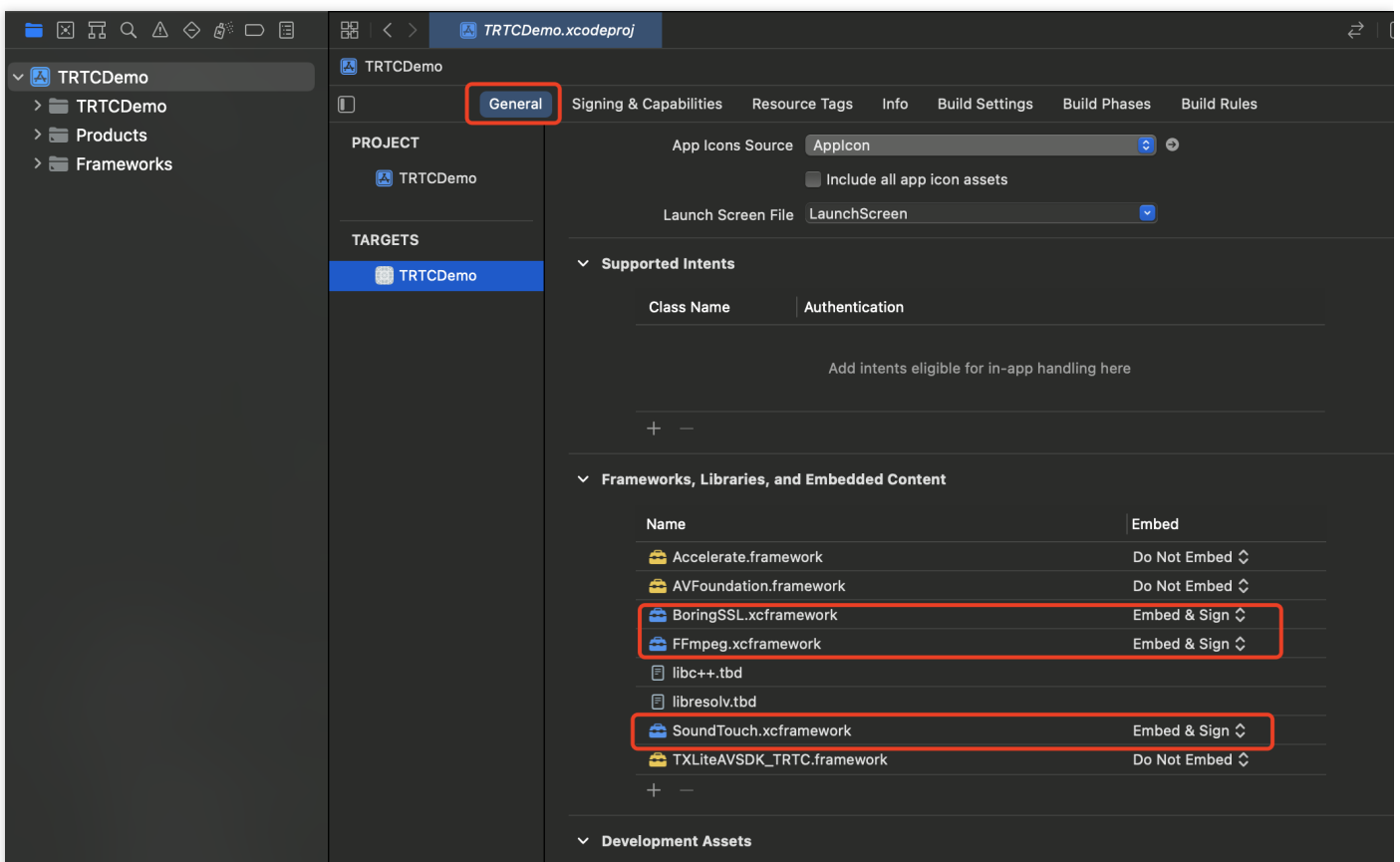


4. Add the downloaded TRTC SDK framework and the dependent libraries it requires: **libc++.tbd**, **Accelerate.framework**, **libresolv.tbd**, and **AVFoundation.framework**.



5. If you use **TRTC SDK v9.5.11234 or a later version**, you need to manually add the dynamic libraries.

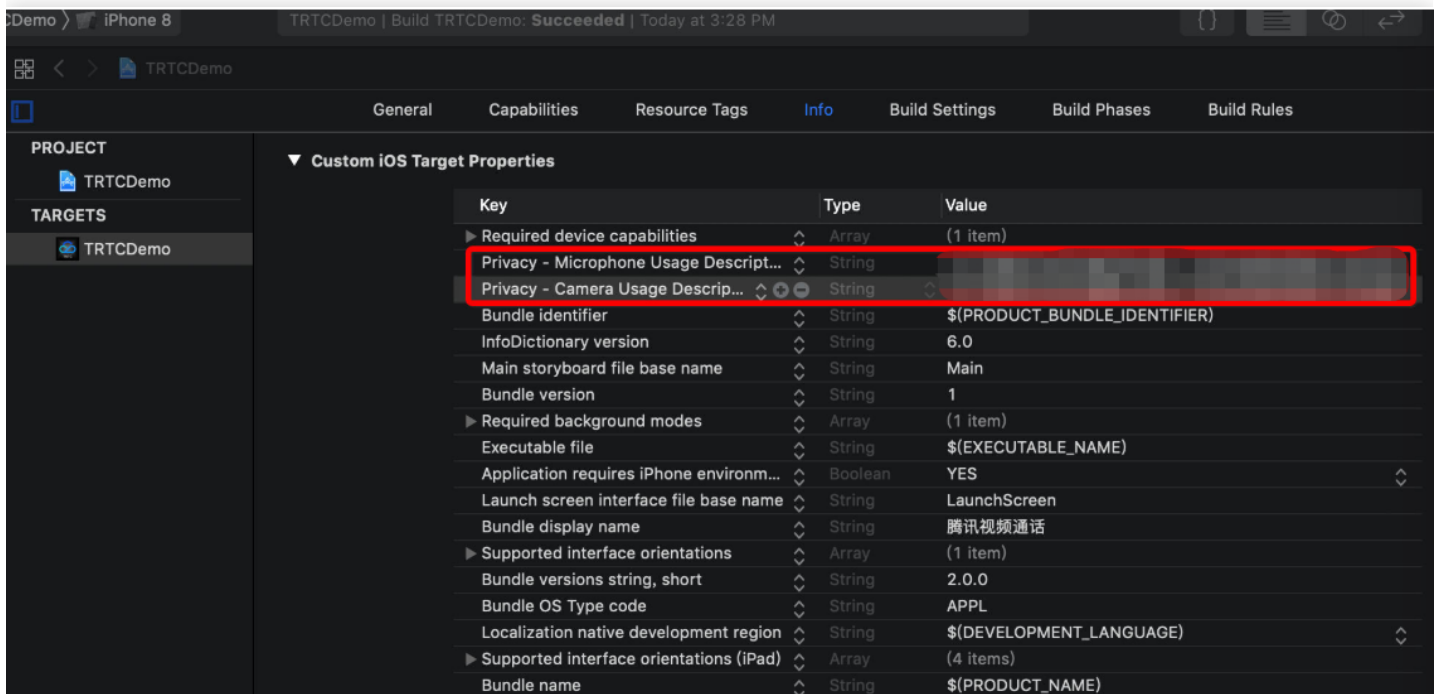
Click **General**, expand **Frameworks, Libraries, and Embedded Content**, and click the + icon at the bottom to add the dynamic libraries required by `TXLiteAVSDK_TRTC.framework` in turn: **BoringSSL.xcframework**, **FFmpeg.xcframework**, and **SoundTouch.xcframework**. Click **Embed & Sign**.



## Granting Camera and Mic Permissions

To use the audio/video features of the SDK, you need to grant it mic and camera permissions. Add the two items below to `Info.plist` of your application. Their content is what users see in the mic and camera access pop-up windows.

- **Privacy - Microphone Usage Description**, plus a statement specifying why mic access is needed
- **Privacy - Camera Usage Description**, plus a statement specifying why camera access is needed



## Importing the TRTC SDK

You can import the TRTC SDK in two ways.

### Method 1: Using Objective-C or Swift APIs

There are two ways to use the SDK in Objective-C or Swift:

- **Import the module:** Import the SDK module in the files that will use the SDK APIs.



```
@import TXLiteAVSDK_TRTC;
```

- **Import the header file:** Import the header file in the files that will use the SDK APIs.

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

## Method 2: Using C++ APIs

1. **Import the header file:** If you want to use C++ APIs to develop your iOS application, import the header file in the `TXLiteAVSDK_TRTC.framework/Headers/cpp_interface` directory.

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

2. **Use the namespace:** The cross-platform C++ methods and types are all defined in the TRTC namespace, which you can use directly. This method can simplify your code and is recommended.

```
using namespace trtc;
```

Note :

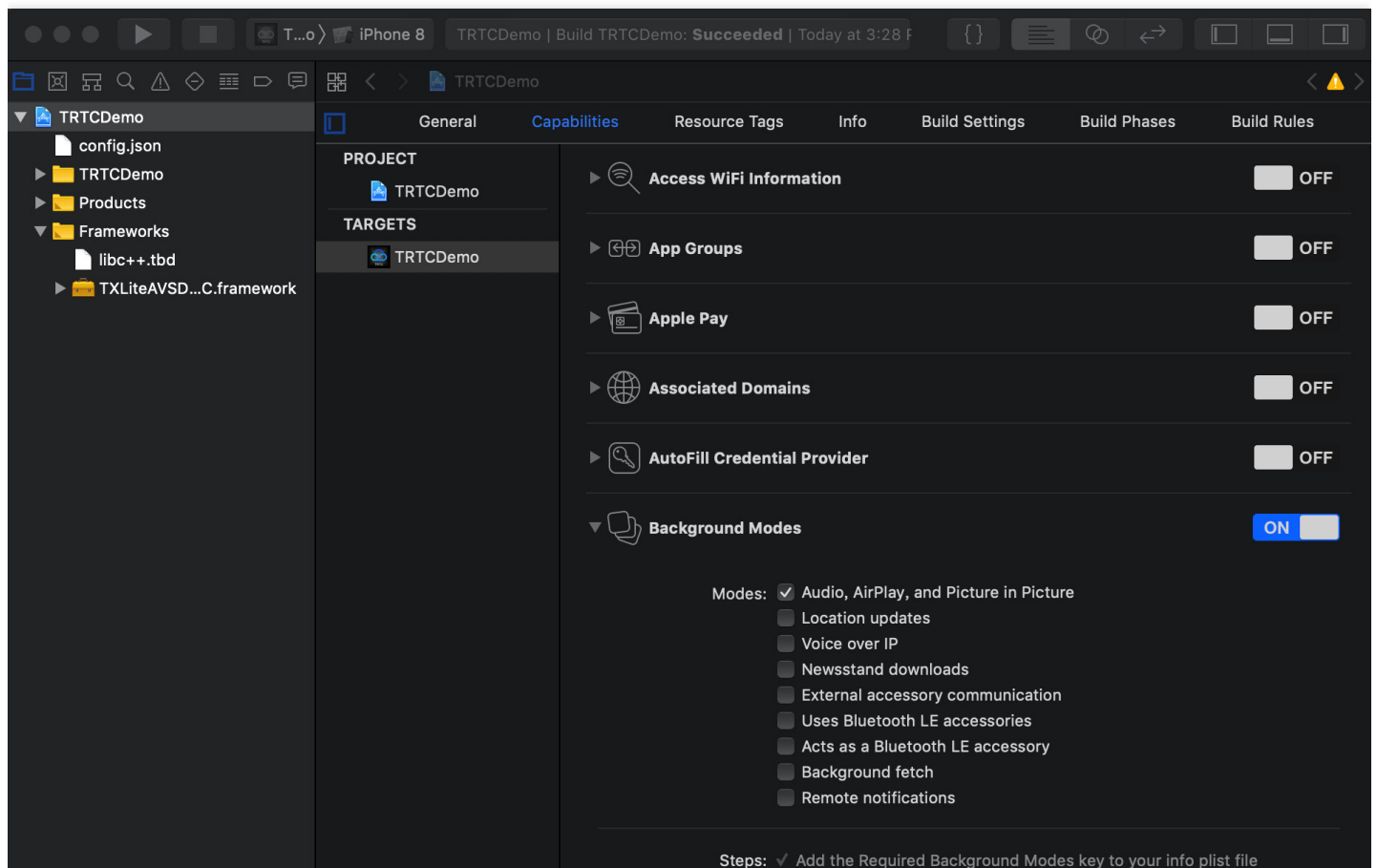
For more information on how to use C++ APIs, please see [Overview](#).

## FAQs

### Can the SDK run in the background?

Yes. If you want the SDK to run in the background, select your project, under the **Capabilities** tab, toggle on **Background Modes**, and select **Audio, AirPlay, and Picture in Picture**, as shown

below:



# Android

Last updated : 2022-03-03 14:39:24

This document describes how to quickly integrate the Tencent Cloud TRTC SDK for Android into your project in the following steps.

## Environment Requirements

- Android Studio 3.5 or above.
- Android 4.1 (SDK API 16) or above.

## Integrating the SDK (AAR)

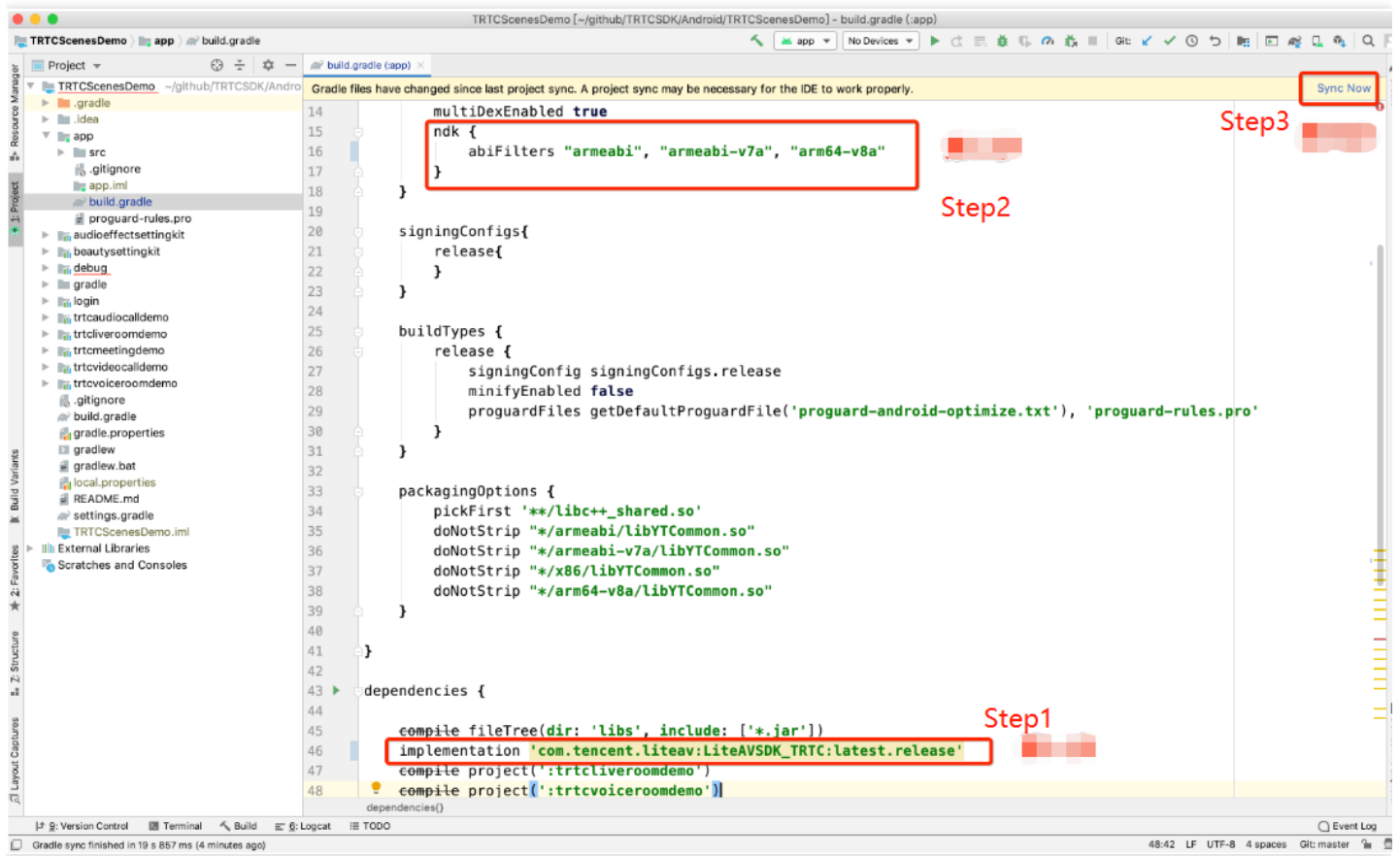
You can use Gradle to automatically load the AAR file or manually download the AAR file and import it into your project.

### Method 1. Automatic loading (aar)

The TRTC SDK has been released to the mavenCentral repository, and you can configure Gradle to download updates automatically.

Simply use Android Studio to open the project that needs to be integrated with the SDK ([TRTCScenesDemo](#) is used as an example in this document), and then modify the `app/build.gradle`

file in three simple steps to complete SDK integration:



1. Add the TRTC SDK dependency to `dependencies`.

- Run the following command to use `com.android.tools.build:gradle v3.x`:

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

- Run the following command if you use the 2.x version of `com.android.tools.build:gradle`.

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

2. In `defaultConfig`, specify the CPU architecture to be used by your application.

```
defaultConfig {
    ndk {
```

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

Note :

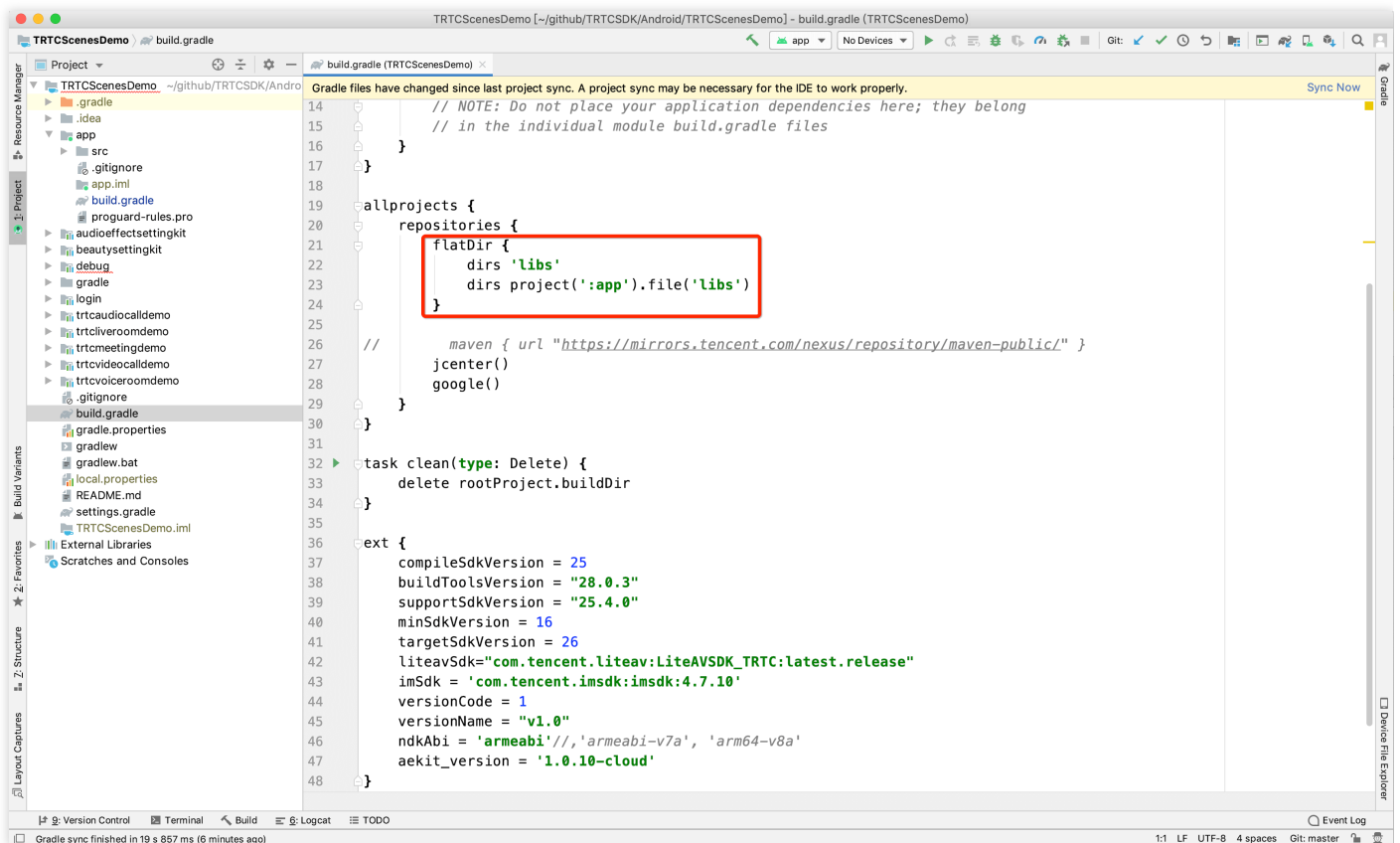
Currently, the TRTC SDK supports armeabi, armeabi-v7a, and arm64-v8a.

3. Click **Sync Now** to automatically download the SDKs and integrate them into your project.

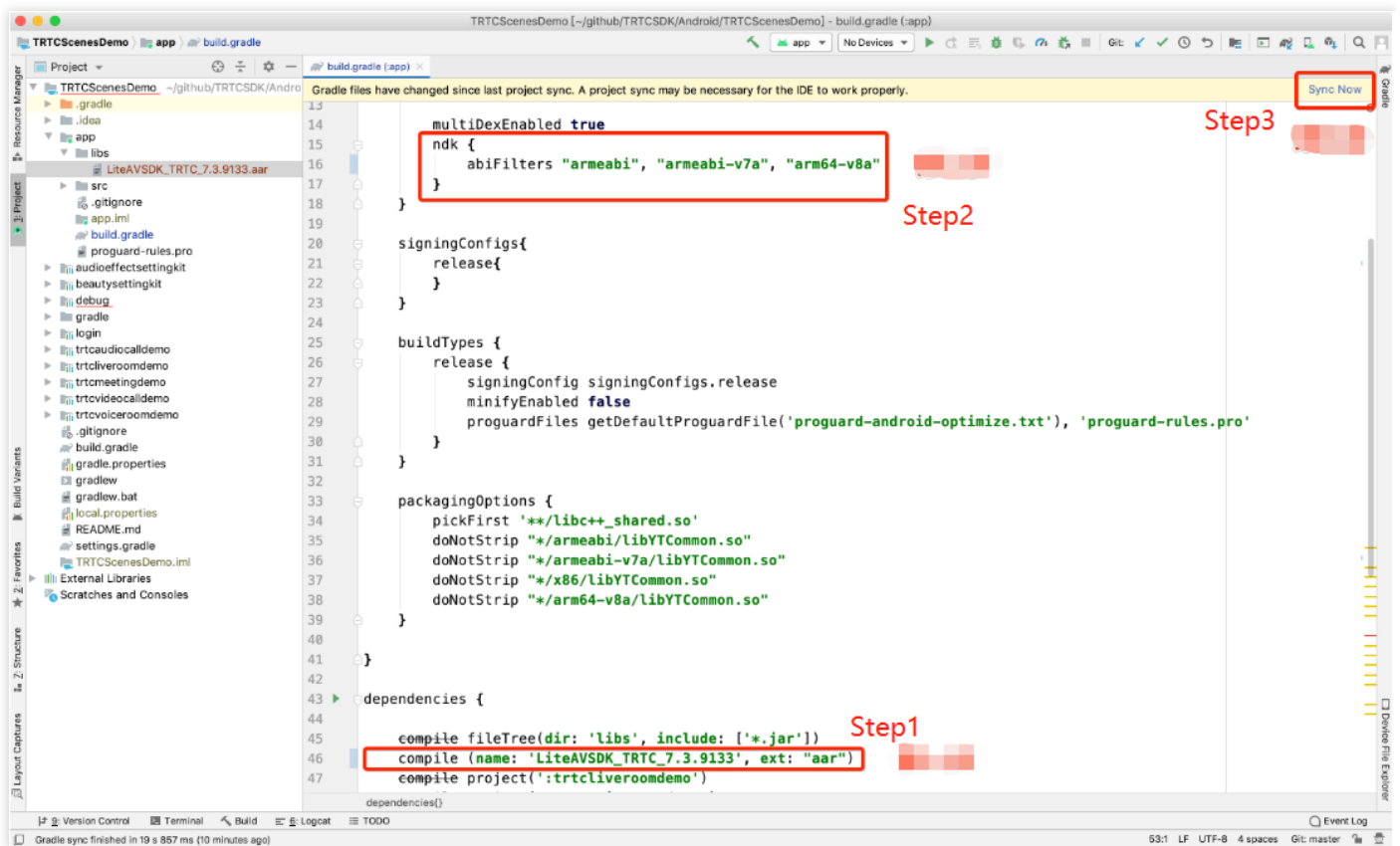
## Method 2: manual download (AAR)

If you have difficulty accessing mavenCentral, you can manually download the SDK and integrate it into your project.

1. [Download the SDK](#).
2. Copy the downloaded AAR file to the **app/libs** directory of your project.
3. Add **flatDir** to `build.gradle` under the project's root directory and specify a local path for the repository.



4. Add code in `app/build.gradle` to import the AAR file.



5. In `defaultConfig` of `app/build.gradle`, specify the CPU architecture to be used by your application.

```

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

```

Note :

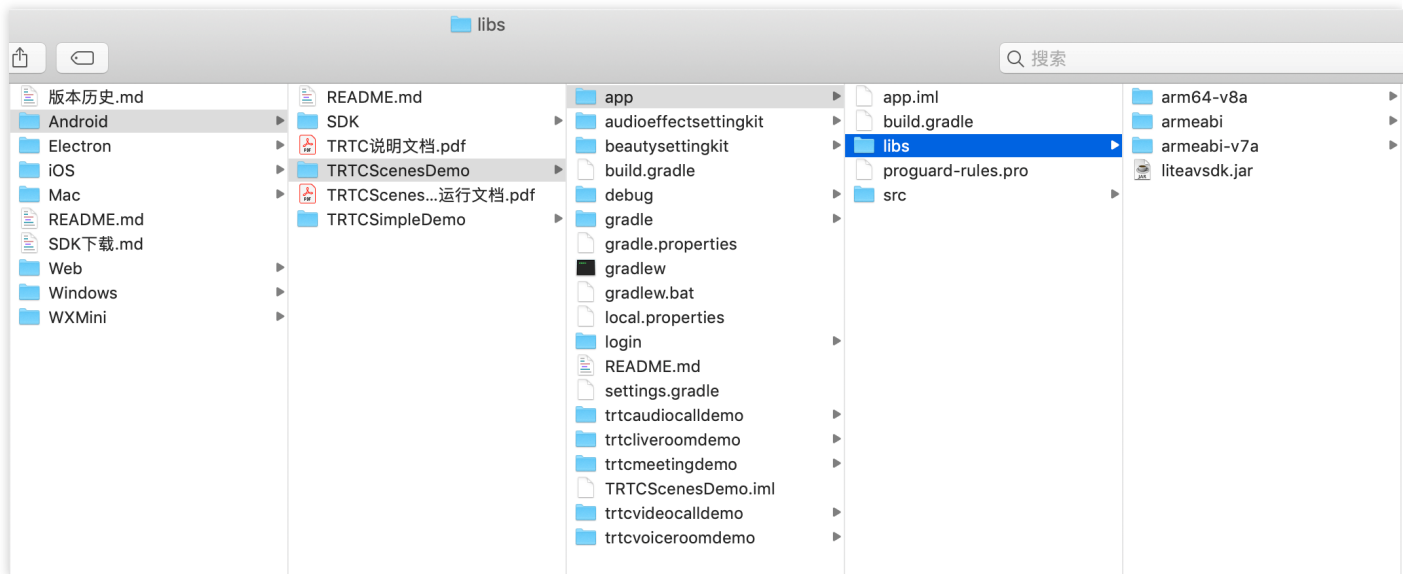
Currently, the TRTC SDK supports armeabi, armeabi-v7a, and arm64-v8a.

6. Click **Sync Now** to complete the integration of TRTC SDK.

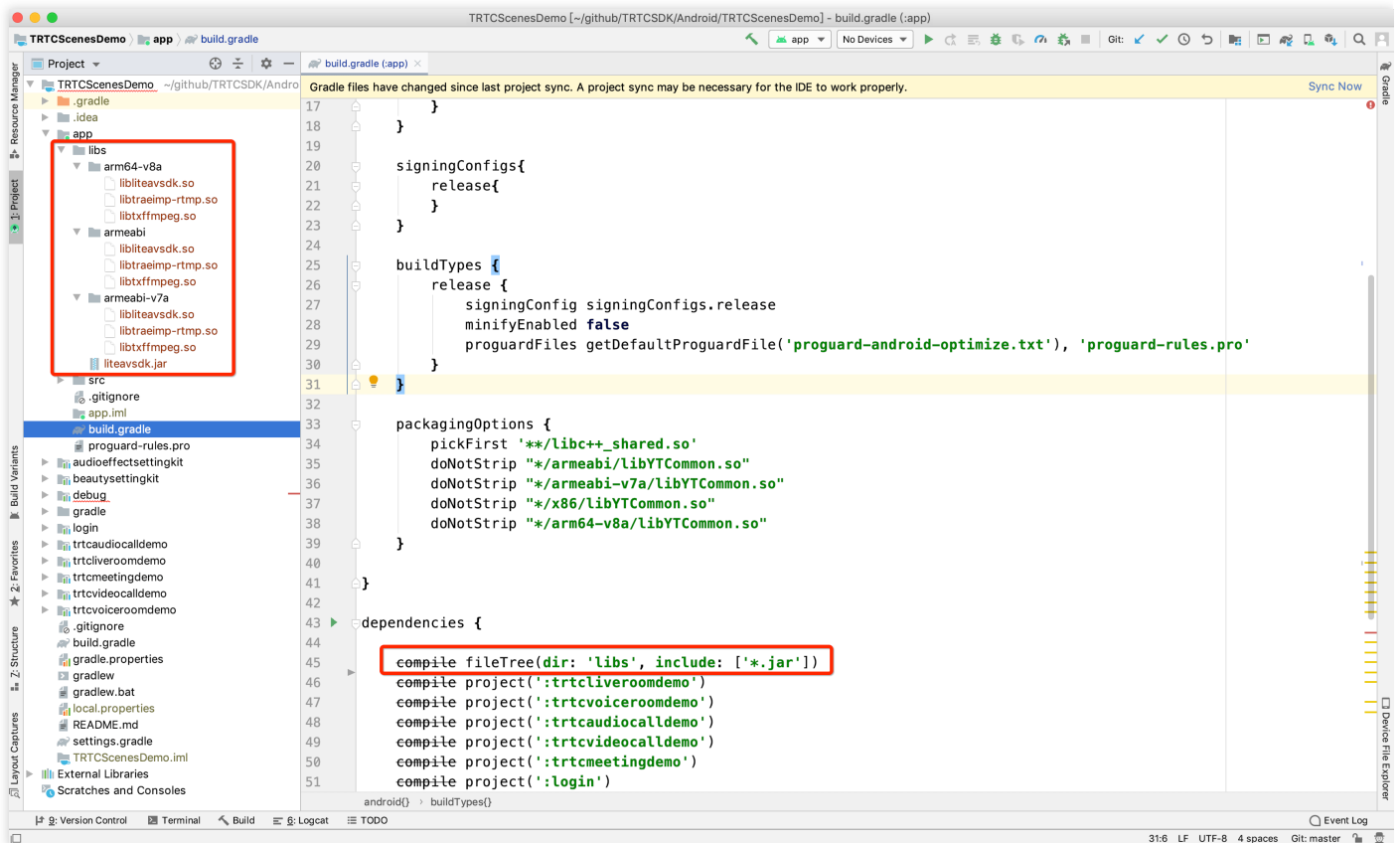
## Integrating SDK (JAR)

If you do not want to import the AAR library, you can also integrate TRTC SDK by importing JAR and SO libraries.

1. [Download the JAR library](#). The file path is `SDK/LiteAVSDK_TRTC_xxx.zip` (xxx indicates the version number of TRTC SDK).
2. Decompress the file, and you will find a `libs` directory that contains a JAR file and several SO folders.
3. Copy the JAR file and `armeabi`, `armeabi-v7a`, and `arm64-v8a` folders to the `app/libs` directory.



4. Add the code that imports the JAR library to `app/build.gradle`.



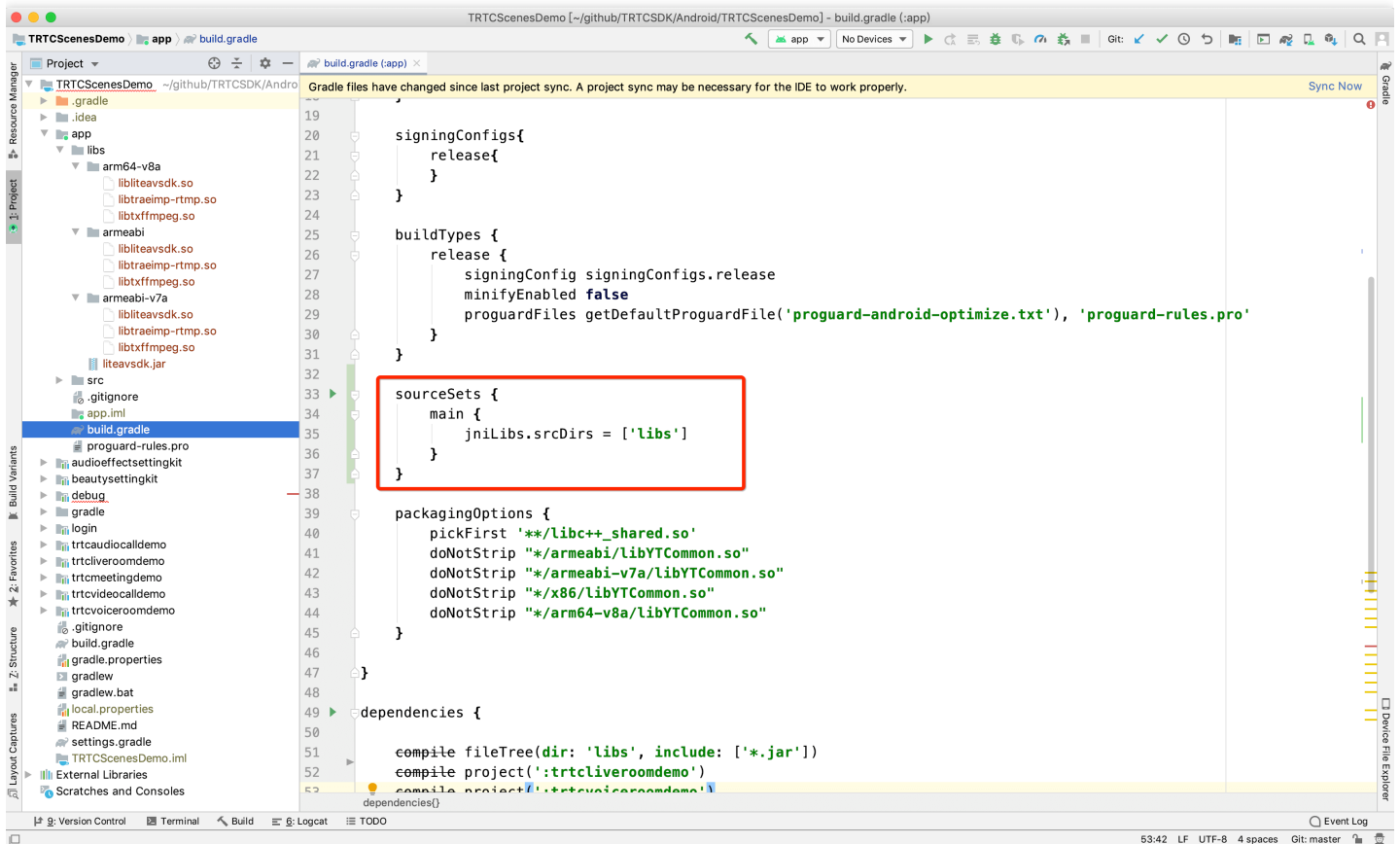
5. Add the code that imports the SO library to `app/build.gradle`.

```

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

```





6. In `defaultConfig` of `app/build.gradle`, specify the CPU architecture to be used by your application.

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

Note :

Currently, the TRTC SDK supports armeabi, armeabi-v7a, and arm64-v8a.

7. Click **Sync Now** to complete the integration of TRTC SDK.

## Configuring Application Permissions

Configure application permissions in `AndroidManifest.xml`. The TRTC SDK requires the following permissions:

```
<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" />
```

Note :

Do not set `android:hardwareAccelerated="false"`; otherwise, the video stream of the remote user cannot be rendered after hardware acceleration is disabled.

## Setting Obfuscation Rules

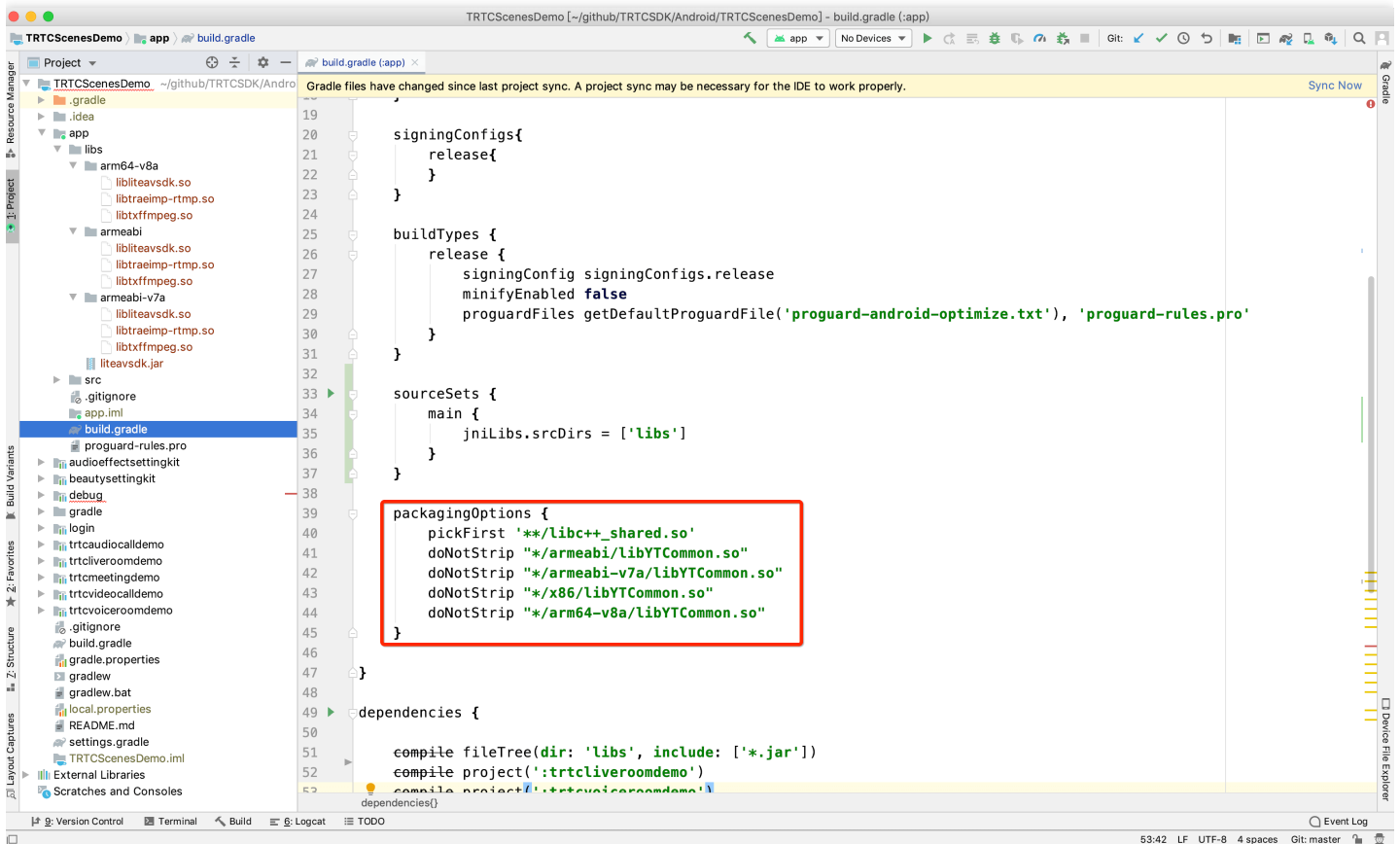
In the `proguard-rules.pro` file, add the classes related to the TRTC SDK to the "do not obfuscate" list:

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

## Setting Application Packaging Parameters

Add the following code to `app/build.gradle` :

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



## Using SDK Through C++ APIs (Optional)

If you prefer to use C++ APIs instead of Java for development, you can perform this step. If you only use Java to call the TRTC SDK, please skip this step.

1. First, you need to integrate the TRTC SDK by importing JAR and SO libraries as instructed above.
2. Copy the C++ header file in the SDK to the project (path: `SDK/LiteAVSDK_TRTC_xxx/libs/include`) and configure the `include` folder path and dynamic link to the SO library in `CMakeLists.txt`.

```

cmake_minimum_required(VERSION 3.6)
# Configure the C++ API header file path
include_directories(
    ${CMAKE_CURRENT_SOURCE_DIR}/include # Copied from `SDK/LiteAVSDK_TRTC_xxx/libs/include`
)
add_library(
    native-lib
    SHARED
    native-lib.cpp

```

```
# Configure the path of the `libliteavsdk.so` dynamic library
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)
# Configure the dynamic link as `libliteavsdk.so`
target_link_libraries(
native-lib
libliteavsdk
${log-lib})
```

3. Use the namespace: the methods and types of cross-platform C++ APIs are defined in the `trtc` namespace. To simplify your code, you are advised to use the `trtc` namespace.

```
using namespace trtc;
```

Note :

- For more information on how to configure the Android Studio C/C++ development environments, please see [Add C and C++ code to your project](#).
- Currently, only the TRTC edition of the SDK supports C++ APIs. For more information on how to use C++ APIs, please see [Overview](#).

# macOS

Last updated : 2022-04-02 17:49:38

This document describes how to quickly integrate the TRTC macOS SDK into your project.

## Environment Requirements

- Xcode 9.0 or above
- A Mac computer with OS X 10.10 or above
- A valid developer signature for your project

## Integrating the TRTC SDK

You can use CocoaPods to automatically load the SDK or download and import it manually into your project.

### CocoaPods

#### 1. Install CocoaPods

Enter the following command in a terminal window (you need to install Ruby on your Mac first):

```
sudo gem install cocoapods
```

#### 2. Create a Podfile

Go to the directory of your project and enter the following command to create a Podfile in the directory.

```
pod init
```

#### 3. Edit the Podfile

There are two ways to edit the Podfile:

- Method 1: Use the pod path of the LiteAV SDK

```
platform :osx, '10.10'  
  
target 'Your Target' do
```

```
pod 'TXLiteAVSDK_TRTC_Mac', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsdkspec/TXLiteAVSDK_TRTC_Mac.podspec'
end
```

- Method 2: Use CocoaPod's official source, which allows version selection

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

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

## 4. Install and update the SDK

Enter the following command in a terminal window to install the SDK.

```
pod install
```

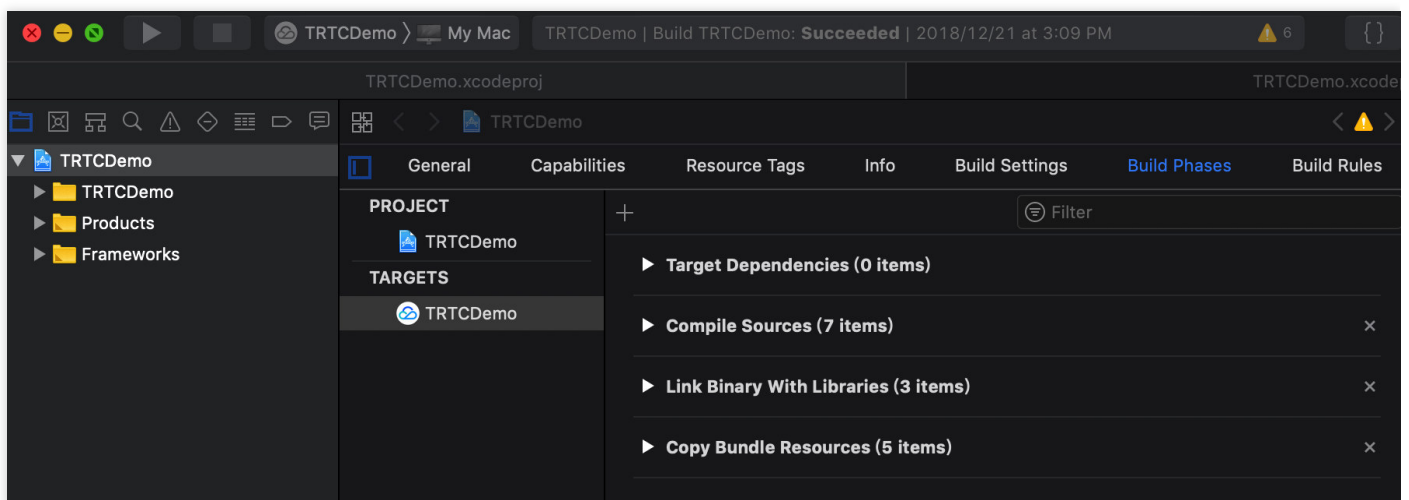
Or, run this command to update the local repository:

```
pod update
```

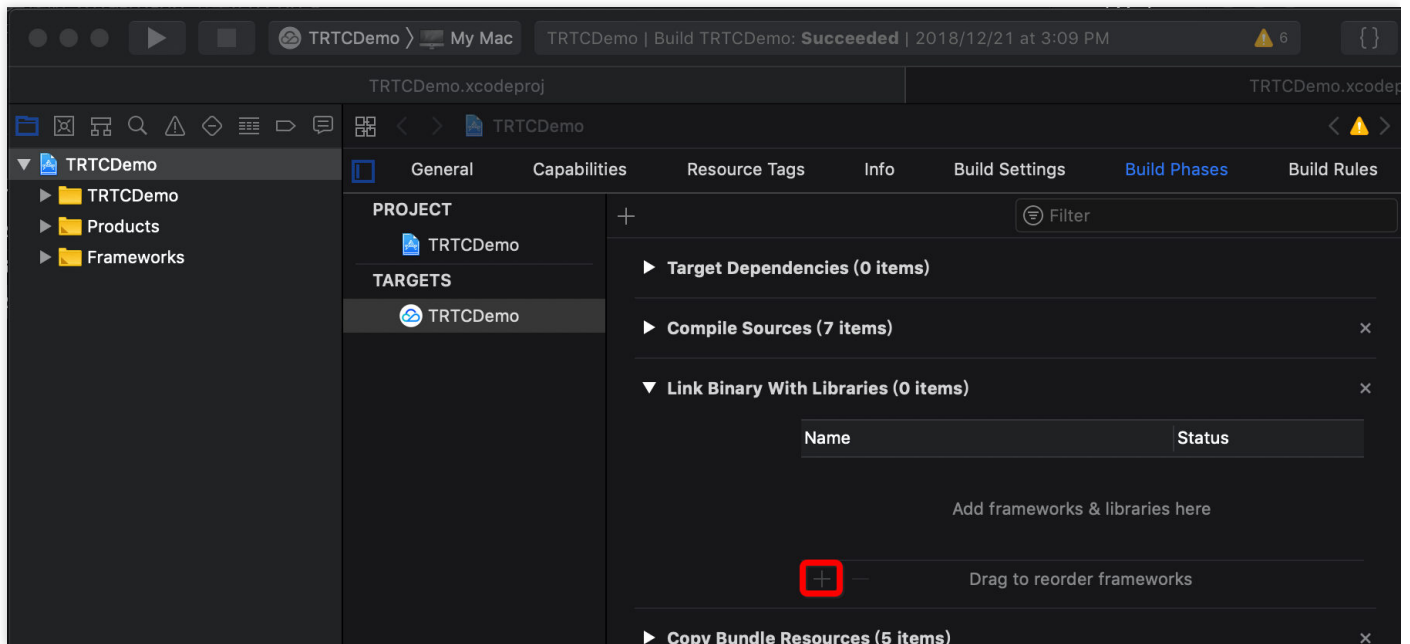
An XCWORKSPACE project file integrated with LiteAVSDK will be generated. Double-click to open the file.

## Manual integration

1. Download the [TRTC macOS SDK](#).
2. Open your Xcode project and import into it the framework downloaded in step 1.
3. Select the target you want to run and click **Build Phases**.

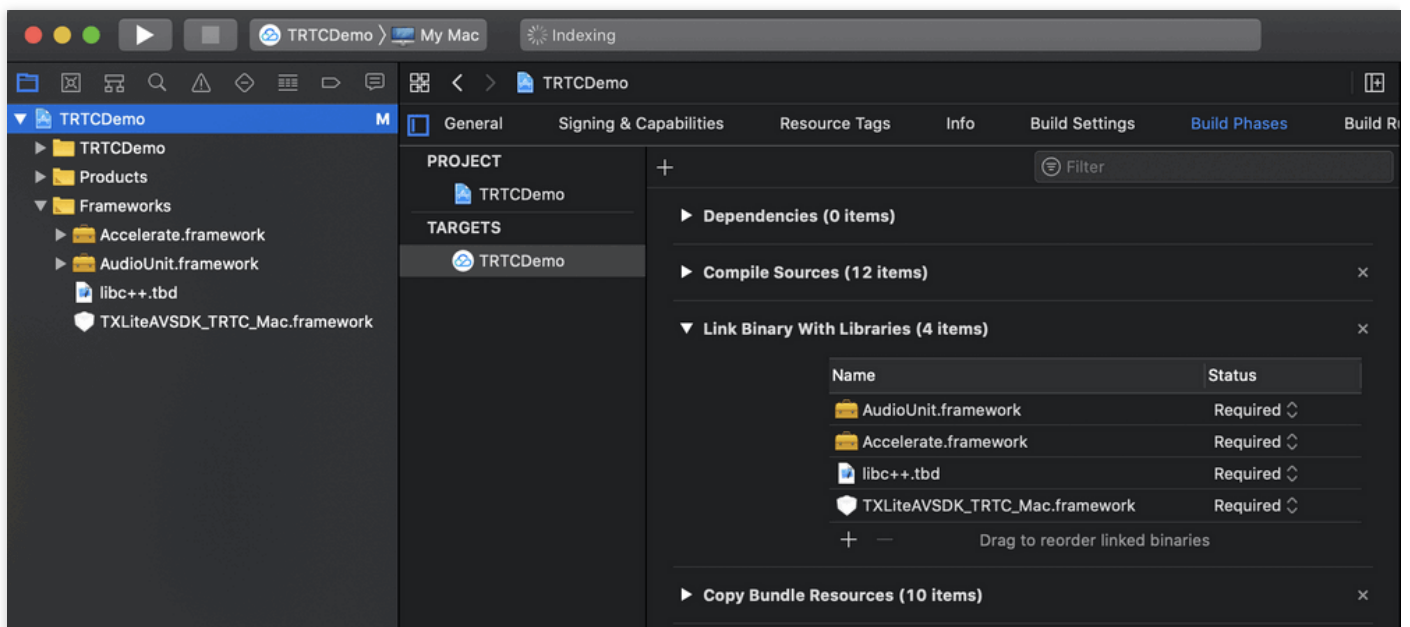


4. Expand **Link Binary With Libraries** and click the **+** icon at the bottom to add dependent libraries.



5. Add the downloaded SDK framework and the dependent libraries it requires: `AudioUnit.framework`, `libc++.tbd`, and `Accelerate.framework`.

After that, you will see:

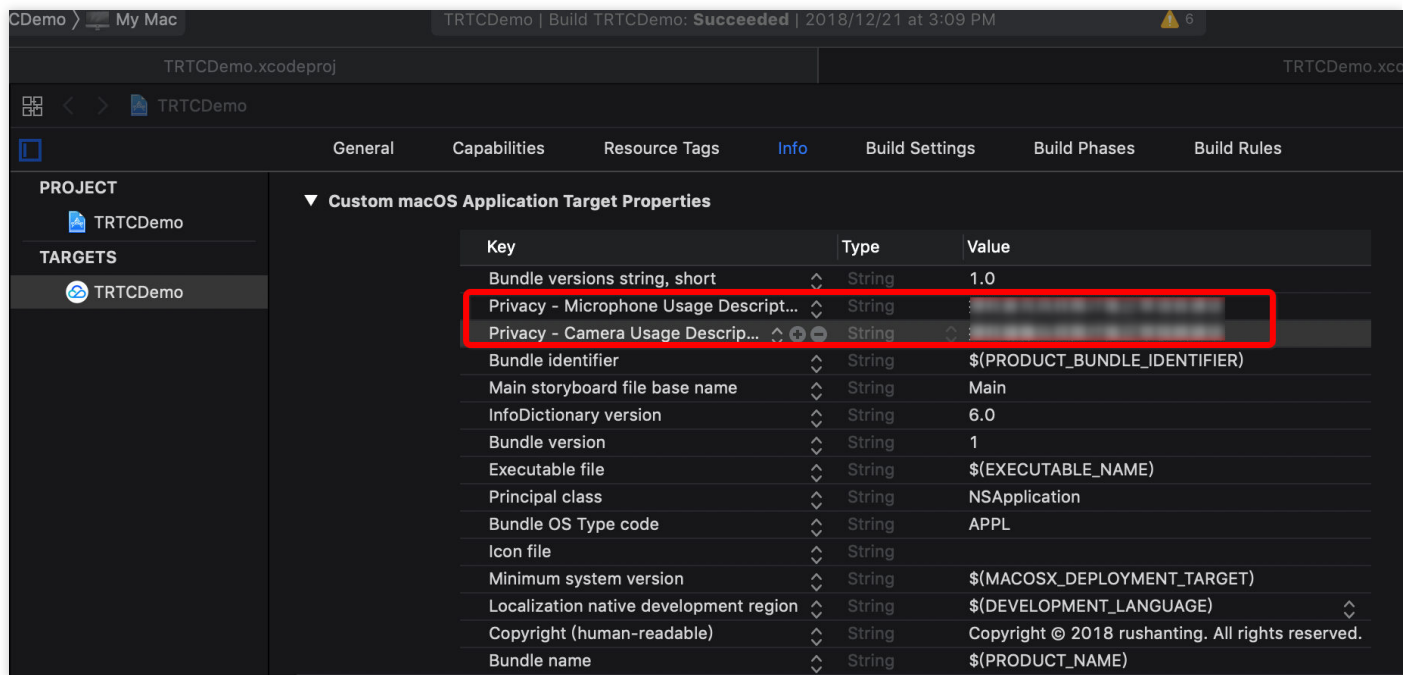


## Granting Camera and Mic Permissions

To use the audio/video features of the SDK, you need to grant it mic and camera permissions. Add the two items below to `Info.plist` of your application. Their content is what users see in the mic and camera access pop-up windows.

- **Privacy - Microphone Usage Description**, plus a statement specifying why mic access is needed
- **Privacy - Camera Usage Description**, plus a statement specifying why camera access is needed

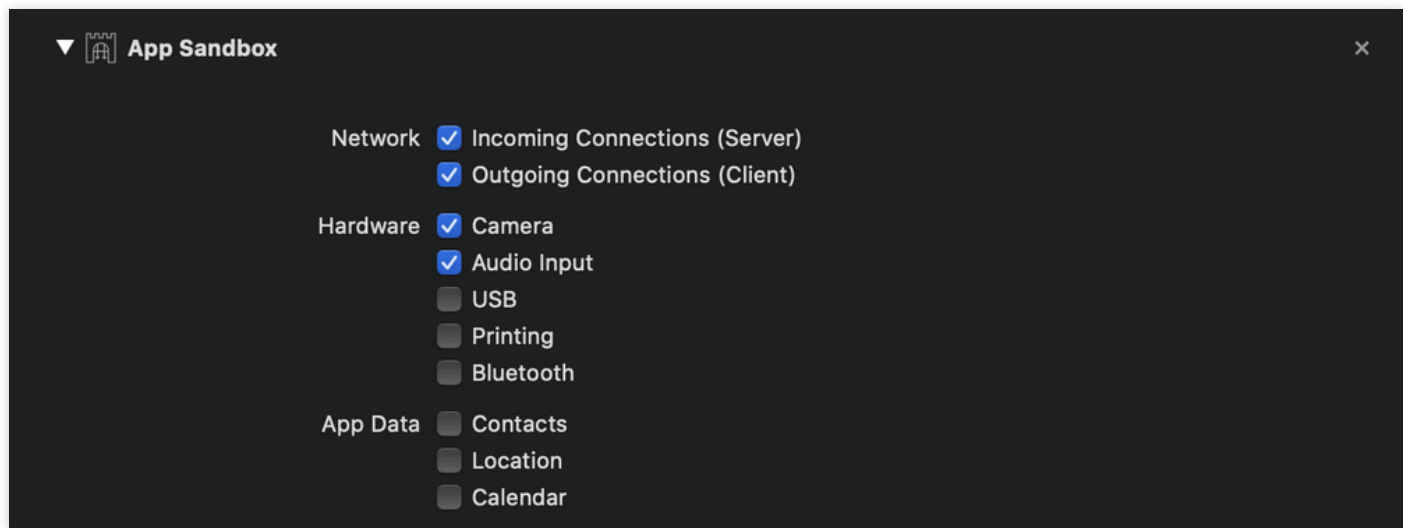
As shown below:



If **App Sandbox** or **Hardened Runtime** is enabled for your application, select `Network` , `Camera` , and `Audio Input` .



- For App Sandbox:



- For Hardened Runtime:

▼ ○ Hardened Runtime

Runtime Exceptions

☐

**Allow Execution of JIT-compiled Code**  
Useful in conjunction with JavaScriptCore.framework or other frameworks relying on JIT compilation. Allows creating writable and executable memory using the MAP\_JIT flag.

☐

**Allow Unsigned Executable Memory**  
Useful for legacy applications that create executable code in memory. Allows creating writable and executable memory without using the MAP\_JIT flag.

☐

**Allow DYLD Environment Variables**  
Allows an application to be impacted by DYLD environment variables, which can be used to inject code into the process.

☐

**Disable Library Validation**  
Allows an application to load plug-ins or frameworks signed by other developers.

☐

**Disable Executable Memory Protection**  
Disables all code signing protections on the application while executing. Useful for legacy applications that modify their own executable code in memory.

☐

**Debugging Tool**  
Declares the application as a debugger. Useful for applications that need to attach to other processes or get task ports.

Resource Access

☒

**Audio Input**  
Allows recording of audio using the built-in microphone, if available, along with access to audio input using any Core Audio API that supports audio input.

☒

**Camera**  
Allows capture of movies and still images using the built-in camera, if available.

☐

**Location**  
Grants access to Location Services location information.

☐

**Contacts**  
Provides read/write access to contacts in the user's address book; allows apps to infer the default address book if more than one is present on a system.

☐

**Calendar**  
Provides read/write access to the user's calendars.

☐

**Photos Library**  
Provides read/write access to the user's Photos library.

☐

**Apple Events**  
Allows posting of AppleEvents to other applications.

# Importing the TRTC SDK

You can import the TRTC SDK in two ways.

## Method 1: Using Objective-C or Swift APIs

There are two ways to use the SDK in Objective-C or Swift:

- **Import the module:** Import the SDK module in the files that will use the SDK APIs.

```
@import TXLiteAVSDK_TRTC_Mac;
```

- **Import the header file:** Import the header file in the files that will use the SDK APIs.

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

## Method 2: Using C++ APIs

1. **Import the header file:** If you want to use C++ APIs to develop your macOS application, import the header file in the `TXLiteAVSDK_TRTC_Mac.framework/Headers/cpp_interface` directory.

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

2. **Use the namespace:** The cross-platform C++ APIs and types are all defined in the TRTC namespace, which you can use directly. This method can simplify your code and is recommended.

```
using namespace trtc;
```

Note :

For more information on how to use C++ APIs, please see [Overview](#).

# Windows

Last updated : 2021-04-12 15:35:17

This document describes how to quickly integrate the Tencent Cloud TRTC SDK for C# on Windows into your project.

## Development Environment Requirements

- OS: Windows 7 or above.
- Development environment: Visual Studio 2010 or above (v2015 is recommended).
- Development framework: .Net Framework 4.0 or above.

## Integrating the TRTC SDK

This document uses the creation of a simple Winform project as an example to describe how to integrate the SDK for C# into a Visual Studio project.

### Step 1. Download the SDK for Windows.

[Download the SDK](#), decompress it, and open the files, including:

Directory Name	Description
xxxDemo	Source code of the demos for C++ and C#
CPlusPlus	SDK library file depended on by the 32/64-bit SDK for C++
CSharp	SDK library file depended on by the 32/64-bit SDK for C#

In the example in this document, you only need to import the C# version of the SDK files into the SDK directory.

### Step 2. Create a project

Open Visual Studio and create a Winform application named `TRTCCSharpDemo`.

### Step 3. Copy the files

Copy the extracted SDK folder to the directory where `TRTCCSharpDemo.csproj` is located.

If you only need the SDK for C#, you can delete the `CPlusPlus` directory under the SDK path.

## Step 4. Modify the project configuration

### Step 4.1. Add references

1. Find **Configuration Manager** in the **Build** directory of Visual Studio and open it.
2. Select **New** from the **Active solution platform** drop-down list and the **New Solution Platform** dialog box will appear.
3. Type or select the new platform and click **OK**.
4. Repeat [substep 2](#) to [substep 3](#) as needed to create solution platforms that need to be supported.
5. Open the folder where the TRTCCSharpDemo project is located and edit the `TRTCCSharpDemo.csproj` file with a text editor.
6. Add the following content to the `<itemGroup>` label in the `TRTCCSharpDemo.csproj` file:

```
// Add references to different platforms
<Reference Include="ManageLiteAV" Condition="'$(Platform)' == 'x64'">
  <HintPath>SDK\CSSharp\Win64\Lib\ManageLiteAV.dll</HintPath>
</Reference>
<Reference Include="ManageLiteAV" Condition="'$(Platform)' == 'AnyCPU'">
  <HintPath>SDK\CSSharp\Win64\Lib\ManageLiteAV.dll</HintPath>
</Reference>
<Reference Include="ManageLiteAV" Condition="'$(Platform)' == 'x86'">
  <HintPath>SDK\CSSharp\Win32\Lib\ManageLiteAV.dll</HintPath>
</Reference>
```

### Step 4.2. Add the copy command

1. Open the properties page of the TRTCCSharpDemo by selecting **Solution Explorer > Right-click Menu of TRTCCSharpDemo Project > Properties**.
2. Add the following command in **Build Events > Post-build event command line** to implement automatic copying of the .dll files of the SDK on different platforms to the run directory of the program after compilation is completed as shown below:

```
set Platform=Win64
SETLOCAL ENABLEDELAYEDEXPANSION
if $(PlatformName)==x86 (
set Platform=Win32
)
copy /Y "$(ProjectDir)SDK\CS\Sharp\!Platform!\Lib\*.dll" "$(ProjectDir)\$(OutDir)"
ENDLOCAL
```

### Step 4.3. Modify the debugging environment

Open the properties page of the TRTCDemo, select **Build**, and set **Platform** to the solution platform in the top menu bar as shown below:

### Step 5. Print the SDK version number

1. Add a label control in the designer of `Form1.cs` as shown below:
2. Open the `Form1.cs` code file and add the following code:

```
using System.Windows.Forms;
using ManageLiteAV; // 1. Add namespace reference

namespace TRTCCSharpDemo
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            // 2. Get an `ITRTCCloud` instance and print the SDK version number
            ITRTCCloud lTRTCCloud = ITRTCCloud.getTRTCShareInstance();
            this.label1.Text = "SDK version : " + lTRTCCloud.getSDKVersion();
            // 3. The `ITRTCCloud` instance needs to be manually terminated at the end of use
            ITRTCCloud.destroyTRTCShareInstance();
        }
    }
}
```

3. Press F5 to run and print the version number of the SDK as shown below:



## FAQs

- If the following error occurs, please check whether the reference to the SDK is added to the project as instructed in [Modify the project configuration](#).

Error CS0246: The type or namespace name "ManageLiteAV" could not be found (are you missing a using directive or an assembly reference?)

- If the following error occurs, please check whether the project's running platform is the current target platform of the solution as instructed in [Modify the project configuration](#).

System.BadImageFormatException: "Could not load file or assembly "ManageLiteAV, Version=2.0.7152.18518, Culture=neutral, PublicKeyToken=null" or one of its dependencies. An attempt was made to load a program with an incorrect format."

- If the following error occurs, please check whether the built event is correctly added to the running directory as instructed in [Modify the project configuration](#).

System.IO.FileNotFoundException: "Could not load file or assembly "ManageLiteAV.dll" or one of its dependencies. The specified module could not be found."

- Due to possible compatibility issues between different Windows versions, the following. dll files have been added to the SDK for C# to solve such issues. File lists as shown below:



# Web

Last updated : 2022-02-14 11:40:52

This document describes how to quickly integrate the Tencent Cloud TRTC SDK for Web into your project.

## Supported Platforms

The TRTC SDK for web is based on WebRTC, which was originally released by Google and is well supported by many modern browsers such as Chrome, Edge, Firefox, Safari, and Opera. For a list of browsers supported by TRTC, see [Supported Platforms](#).

- If your application scenario is mainly in the education sector, consider using [TRTC SDK for Electron](#), which supports big and small (dual-channel) images, with more flexible screen sharing schemes and better recovery capabilities for poor network connections.

Note :

- You can run the [TRTC Web SDK Support Level Test](#) in a browser, for example, WebView, to test whether the environment fully supports WebRTC.
- Due to patent issues, H.264 encoding, which is required for stream publishing, is unavailable for Chrome versions earlier than v88 on Huawei devices. To run the TRTC SDK for web on Chrome or Chrome WebView-based browsers on Huawei devices, please [submit a ticket](#) to enable VP8 encoding/decoding.

## URL Protocol Support

Scenario	Protocol	Receive (Playback)	Send (Publish)	Share Screen	Remarks
Production	HTTPS	Supported	Supported	Supported	Recommended
Commercial	HTTP	Supported	Not supported	Not supported	
Local development	http://localhost	Supported	Supported	Supported	Recommended

Scenario	Protocol	Receive (Playback)	Send (Publish)	Share Screen	Remarks
Local development	http://127.0.0.1	Supported	Supported	Supported	
Local development	http://[local IP address]	Supported	Not supported	Not supported	
Local development	file:///	Supported	Supported	Supported	

## Firewall Configuration

The TRTC SDK for Web uses the following ports for data transfer, which should be added to the whitelist of the firewall.

- TCP port: 8687
- UDP ports: 8000, 8080, 8800, 843, 443, 16285
- Domain name: qcloud.rtc.qq.com

## Integrating the TRTC SDK for Web

### Integrating via npm

1. Use npm to install the SDK package in your project.

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

2. Import the module in the project script.

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

### Integrating via script

Add the following code to your webpage:

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

## Resources

Download the SDK [here](#).

For more information on the initialization process and how to use APIs, please see the tutorials below:

Feature	Sample Code
Audio/Video call	<a href="#">Tutorial</a>
Interactive live streaming	<a href="#">Tutorial</a>
Switching cameras/mics	<a href="#">Tutorial</a>
Setting local video attributes	<a href="#">Tutorial</a>
Dynamically enabling/disabling local audio/video	<a href="#">Tutorial</a>
Screen sharing	<a href="#">Tutorial</a>
Detecting volume	<a href="#">Tutorial</a>
Custom capturing and rendering	<a href="#">Tutorial</a>
Limit on the number of upstream users in a room	<a href="#">Tutorial</a>
Adding background music and audio effects	<a href="#">Tutorial</a>
Environment and device check before calls	<a href="#">Tutorial</a>
Network quality check before calls	<a href="#">Tutorial</a>
Device plugging/unplugging check	<a href="#">Tutorial</a>
Publishing to CDN	<a href="#">Tutorial</a>
Enabling dual-channel mode	<a href="#">Tutorial</a>
Enabling beauty filters	<a href="#">Tutorial</a>
Enabling watermarking	<a href="#">Tutorial</a>
Enabling cross-room communication	<a href="#">Tutorial</a>

Note :

Learn more about the features of the TRTC SDK for web [here](#).

# Electron

Last updated : 2022-01-19 15:23:14

This document describes how to quickly integrate Tencent Cloud TRTC SDK for Electron into your project.

## Supported Platforms

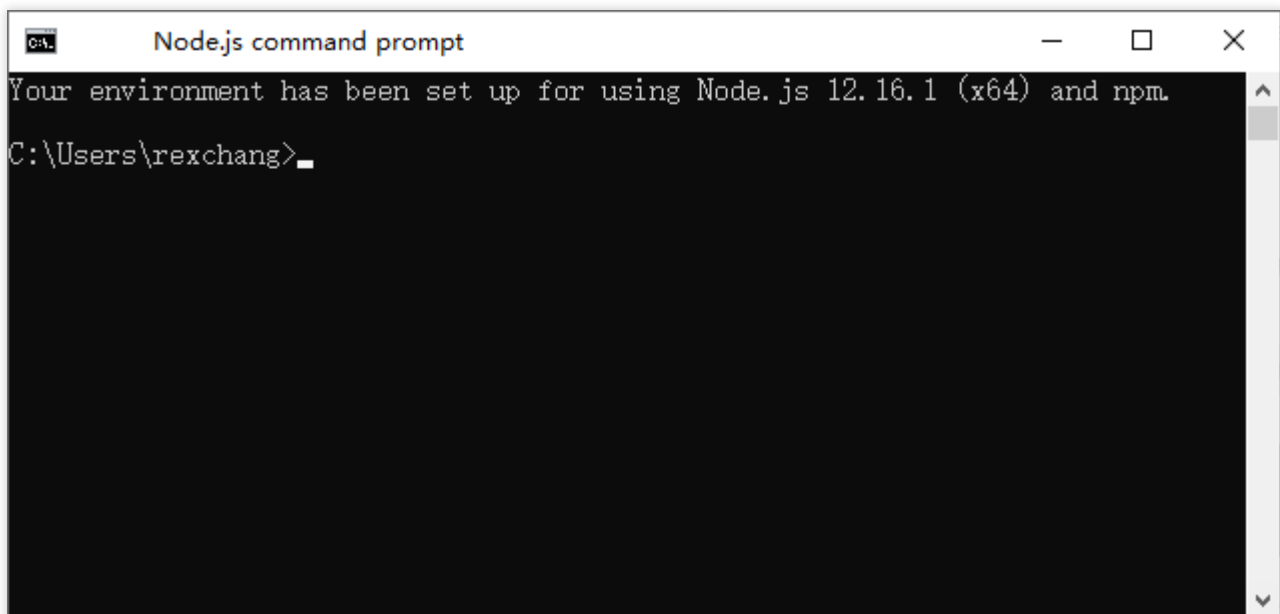
- Windows
- macOS

## Integrating TRTC SDK for Electron

### Step 1: Install Node.js

- Windows
- macOS

1. Download the latest version of [Node.js](#) installer `Windows Installer (.msi) 64-bit`.
2. Open `Node.js command prompt` in the application list.



### Step 2. Install Electron

Run the following command in the command prompt to install Electron. Version 4.0.0 or above is recommended.

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

### Step 3. Install the TRTC SDK for Electron

1. Use the following npm command in your Electron project to install the SDK.

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

Note :

You can view the information of the latest version of TRTC SDK for Electron [here](#).

2. Import the module into the project script and use the module.

```
const TRTCCloud = require('trtc-electron-sdk').default;  
// import TRTCCloud from 'trtc-electron-sdk';  
this.rtcCloud = new TRTCCloud();  
// Get the SDK version number  
this.rtcCloud.getSDKVersion();
```

Since v7.9.348, the TRTC SDK for Electron has added the `trtc.d.ts` file to facilitate developers to use TypeScript:

```
import TRTCCloud from 'trtc-electron-sdk';  
const rtcCloud: TRTCCloud = new TRTCCloud();  
// Get the SDK version number  
rtcCloud.getSDKVersion();
```

## Packaging the Executable Program

### Step 1. Install a packaging tool

1. We recommend that you use the packaging tool `electron-builder`. You can run the following command to install it.

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

2. To package TRTC SDK for Electron, i.e., the `trtc_electron_sdk.node` file correctly, you must also run the following command to install `native-ext-loader`.

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

## Step 2. Modify `webpack.config.js`

The `webpack.config.js` file contains the configuration information for project building. You can locate it in the following ways.

- Normally, `webpack.config.js` is in the root directory of the project.
- If you create your project with `create-react-app`, the configuration file will be `node_modules/react-scripts/config/webpack.config.js`.
- If you create your project with `vue-cli`, webpack configuration will be stored in the `configureWebpack` property of `vue.config.js`.
- If your project is customized, please locate webpack configuration by yourself.

1. First, `webpack.config.js` must receive the `--target_platform` command line parameter so that your project can be packaged correctly for its target platform. Add the following code before `module.exports`.

```
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)) target = os.platform();
  return target;
})();
```

Note :

In the result returned by `os.platform()`, `darwin` means macOS, and `win32` means Windows (64-bit or 32-bit).

2. Add the following configuration to the `rules` option. The `targetPlatform` variable allows `rewritePath` to switch configurations according to the target platform.

```
rules: [  
  {  
    test: /%.node$/,  
    loader: 'native-ext-loader',  
    options: {  
      rewritePath: targetPlatform === 'win32' ? './resources' : '../Resources'  
    }  
  },  
]
```

This above configuration means:

- If you create an EXE file for Windows, `native-ext-loader` will load the TRTC SDK in `[application root directory]/resources`.
- If you create a DMG file for macOS, `native-ext-loader` will load the TRTC SDK in `[application directory]/Contents/Frameworks/../Resources`.

You also need to add the `--target_platform` parameter to the build script of `package.json`. See step 3 for details.

### Step 3. Modify `package.json`

The `package.json` file is in the root directory of the project and contains information needed for packaging. Normally, to successfully package your project, you need to modify the path in `package.json` as follows.

1. Modify `main`.

```
// In most cases, the name of the `main` file can be customized. For example, in TRTCSimpleDemo, `main` can be configured as:  
"main": "main.electron.js",  
  
// However, for projects created with the `create-react-app` scaffolding tool, `main` must be configured as:  
"main": "public/electron.js",
```

2. Copy the following `build` configuration to your `package.json` file for `electron-builder` to read.

```
"build": {  
  "appId": "[Custom appId]",
```



```

"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. Add command scripts for building and packaging under `scripts` .

The following command scripts are for projects created with `create-react-app` and `vue-cli` . They provide samples for projects created with other tools too.

```

// Use this configuration for projects created with `create-react-app`.
"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 && npm run compile:mac",
  "pack:win64": "npm run build:win && npm run compile:win64"
}

// Use this configuration for projects created with `vue-cli`.
"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 && npm run compile:mac",
  "pack:win64": "npm run build:win && npm run compile:win64"
}

```

Parameter	Description
-----------	-------------

main	Entry point file of Electron, which can be customized in most cases. However, if your project is created with <code>create-react-app</code> , the file must be <code>public/electron.js</code> .
build.win.extraFiles	When packaging for Windows, <code>electron-builder</code> will copy all files in the directory specified by <code>from</code> to <code>bin/win-unpacked/resources</code> (all in lowercase).
build.mac.extraFiles	When packaging for macOS, <code>electron-builder</code> will copy the <code>trtc_electron_sdk.node</code> file specified by <code>from</code> to <code>bin/mac/your-app-name.app/Contents/Resources</code> (capitalize the first letter of each word)
build.directories.output	Output path for packaging. In the sample above, the output file is saved to <code>bin</code> . You can modify it as needed.
build.scripts.build:mac	Script for building for macOS
build.scripts.build:win	Script for building for Windows
build.scripts.compile:mac	Compile into a DMG file for macOS
build.scripts.compile:win64	Compile into an EXE file for Windows
build.scripts.pack:mac	Call <code>build:mac</code> to build the project and then <code>compile:mac</code> to package it into a DMG file
build.scripts.pack:win64	Call <code>build:win</code> to build the project and then <code>compile:win64</code> to package it into an EXE file

## Step 4. Run the packaging command

- Package the project into a DMG file for macOS:

```
$ cd [Project directory]
$ npm run pack:mac
```

The packaging tool will generate an installation file named `bin/your-app-name-0.1.0.dmg`. Publish this file.

- Package the project into an EXE file for Windows:

```
$ cd [Project directory]
$ npm run pack:win64
```

The packaging tool will generate an installation file named `bin/your-app-name Setup 0.1.0.exe`. Publish this file.

Note :

Currently, the TRTC SDK for Electron does not support cross-platform packaging. This means you cannot package your project into an EXE file on macOS or a DMG file on Windows, but we are working on this and may make it possible in the future.

## FAQs

### 1. What are firewall restrictions does the SDK face?

The SDK uses the UDP protocol for audio/video transmission and therefore cannot be used in office networks that block UDP. If you encounter such a problem, see [How to Deal with Firewall Restrictions](#).

### 2. What should I do if an exception occurs when I install or package the TRTC SDK for Electron?

If an exception occurs when you integrate the TRTC SDK for Electron, for example, if installation times out or the `trtc_electron_sdk.node` file fails to load after packaging, you can [contact us](#) for help.

## References

- [SDK API Guide](#)
- [SDK Update Log](#)
- [Simple Demo Source Code](#)
- [API Example Source Code](#)
- [FAQs](#)

# Flutter

Last updated : 2022-04-06 17:10:43

This document describes how to quickly integrate TRTC SDK for Flutter into your project.

Note :

At present, the Flutter SDK only supports Android and iOS

## Environment Requirements

- Flutter 2.0 or above
- Developing for Android:
  - Android Studio 3.5 or above
  - Devices with Android 4.1 or above
- Developing for iOS:
  - Xcode 11.0 or above
  - Your project has a valid developer signature.

## Integrating the SDK

The SDK for Flutter has been published on [Pub](#). You can have the SDK downloaded and updated automatically by configuring `pubspec.yaml`.

1. Add the following dependency to `pubspec.yaml` of your project.

```
dependencies:  
  tencent_trtc_cloud: latest version number
```

2. Grant **camera** and **mic** permissions to enable the audio and video call features.

## iOS

1. Add requests for camera and mic permissions in `Info.plist` :

```
<key>NSCameraUsageDescription</key>  
<string>Video calls are possible only with camera permission.</string>
```

```
<key>NSMicrophoneUsageDescription</key>
<string>Audio calls are possible only with mic permission.</string>
```

2. Add the field `io.flutter.embedded_views_preview` and set the value to `Yes`.

## Android

1. Open `/android/app/src/main/AndroidManifest.xml`.
2. Add `xmlns:tools="http://schemas.android.com/tools"` to "manifest".
3. Add `tools:replace="android:label"` to "application".

Note :

Without the above steps, the ["Android Manifest merge failed"](#) error will occur and the compilation will fail.

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

## FAQs

- [What should I do if the iOS app crashes when I build and run it?](#)
- [What should I do if videos do not show on iOS but do on Android?](#)
- [What should I do if an error occurs when I run CocoaPods on iOS after updating to the latest version of the SDK?](#)
- [What should I do if Android Studio fails to build my project with the error "Manifest merge failed"?](#)

- [What should I do if an error occurs due to the absence of signatures when I debug my project on a real device?](#)
- [Why can't I find the corresponding file after deleting/adding content in the swift file of the plugin?](#)
- [What should I do if the error "Info.plist, error: No value at that key path or invalid key path: NSBonjourServices" occurs when I run my project?](#)
- [What should I do if an error occurs when I run pod install?](#)
- [What should I do if a dependency error occurs when I run my project on iOS?](#)

# Qt Creator

Last updated : 2021-07-29 18:39:57

This document describes how to quickly integrate the Tencent Cloud TRTC SDK into your project using Qt on macOS and Windows.

## Integration on macOS

### Environment Requirements

- OS: OS X 10.10 or above
- Development environment: Qt Creator 4.10.3 or above (Qt Creator 4.13.3 or above is recommended.)
- Development framework: Qt 5.10 or above

### Directions

This section uses a QtTest project created from scratch as an example to show you how to integrate TRTC SDK for C++ into your project in Qt Creator.

#### Step 1. Download TRTC SDK for C++.

1. Download the [SDK](#), and decompress and open the file.
2. Create an empty folder for the SDK in the directory of your QtTest project, and copy `TXLiteAVSDKTRTCMacx.x.x/SDK/TXLiteAVSDKTRTC_Mac.framework` from the package downloaded in step 1 to the folder.

#### Step 2. Configure QtTest.pro.

Go to the directory of your QtTest project, open `QtTest.pro` with a text editor, and add the following SDK references.

```
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
```

### Step 3. Request camera and mic permissions.

For the TRTC SDK to use your camera and mic, you need to add permission requests to `Info.plist`.

```
NSMicrophoneUsageDescription: requesting mic access
NSCameraUsageDescription: requesting camera access
```

This is how your project looks like after the adding of permission requests:

### Step 4. Reference the SDK.

1. Reference the SDK using the header file `#include "ITRTCCloud.h"`.
2. Use the namespace: the methods and types of cross-platform C++ APIs are defined in the `trtc` namespace. To simplify your code, you are advised to use the `trtc` namespace.

Note :

This concludes the integration process, and you can proceed to compile your project. You can download [QTDemo](#) to get more information on the use of cross-platform APIs of the SDK.

## Integration on Windows

### Environment Requirements

- OS: Windows 7 or above
- Development environment: Visual Studio 2015 or above. Visual Studio 2015 is recommended, provided that you have set up a Qt development environment for Visual Studio.

Note :

If you are not sure about how to set up a Qt development environment for Visual Studio, see the second part of [README](#).

### Directions

This section uses a simple QtTest project as an example to show you how to integrate TRTC SDK for C++ into your Visual Studio project.

#### Step 1. Download TRTC SDK for C++.



1. Download the [SDK](#), and decompress and open the file.
2. Create an empty folder for the SDK in the directory of your QtTest project, and copy `TXLiteAVSDKTRTCWin_latest/SDK/CPlusPlus` from the package downloaded in step 1 to the folder.

## Step 2. Configure dependent environment for the QtTest project.

### Scenario 1: using Qt Creator

Go to the directory of the QtTest project, open `QTTest.pro` (created using Qt Creator) with a text editor ([Sublime Text](#) is recommended), and add the following SDK references.

```
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
}
}
```

### Scenario 2: using Visual Studio

If your project is a full-fledged Visual Studio project, you can also configure the SDK library path dependency in Visual Studio in **Properties > Linker > Input and General**, and configure the SDK header file path dependency in **Properties > C/C++ > General**.

## Step 3. Copy files.

When you open `QTTest.pro` with Visual Studio, a `debug/release` folder will be generated automatically in the project directory. You need to copy all the DLL files in `SDK/CPlusPlus/Win32/lib` to the `debug/release` folder.

#### Step 4. Reference the SDK.

1. Reference the SDK using the header file `#include "ITRTCCloud.h"` .
2. Use the namespace: the methods and types of cross-platform C++ APIs are defined in the `trtc` namespace. To simplify your code, you are advised to use the `trtc` namespace.

Note :

This concludes the integration process, and you can proceed to compile your project. You can download [QTDemo](#). to get more information on the use of cross-platform APIs of the SDK.

# Unity

Last updated : 2021-09-29 15:49:37

This document describes how to quickly integrate TRTC SDK for Unity into your project.

## Environment Requirements

- Unity 2020.2.1f1c1 is recommended.
- Supported platforms: Android, iOS, Windows, macOS (alpha testing)
- Modules required: `Android Build Support` , `iOS Build Support` , `Windows Build Support` , `MacOS Build Support`
- If you are developing for iOS, you also need:
  - Xcode 11.0 or above
  - A valid developer signature for your project

## Integrating SDK

1. Download the SDK and [demo source code](#).
2. Decompress the ZIP file and copy `TRTCUnitySDK/Assets/TRTCSDK/SDK` to the `Assets` directory of your project.

## FAQs

### What should I do if a network access error occurs on Android?

Copy `/Assets/Plugins/AndroidManifest.xml` to the same directory of your project.

### What should I do if the SDK does not have mic or camera access on Android?

You need to add mic and camera permission requests manually when building for Android. For details, see the code below:

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

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

# React Native

Last updated : 2021-12-31 15:27:57

This document describes how to quickly integrate the TRTC SDK for React Native into your project.

## Environment Requirements

- React Native 0.63 or above
- Node (above v12) & Watchman
- **Developing for Android:**
  - Android Studio 3.5 or above
  - Devices with Android 4.1 or above
  - Java Development Kit
- **Developing for iOS and macOS:**
  - Xcode 11.0 or above
  - OS X 10.11 or above
  - A valid developer signature for your project
- For how to set up the environment, see the React Native [official document](#).

## Integrating SDK

We have released the TRTC SDK for React Native to [npm](#). You can configure `package.json` to install the SDK.

1. Add the following dependencies to `package.json` of your project:

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

2. Grant **camera** and **mic** permissions to enable the audio and video call features.
  - Android
  - iOS
  - i. Configure application permissions in `AndroidManifest.xml`. The TRTC SDK requires the following permissions:

```
<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" />
```

#### Note

Do not set `android:hardwareAccelerated="false"` . Disabling hardware acceleration will result in failure to render remote users' videos.

#### ii. Manually configure audio and video permission requests.

```
if (Platform.OS === 'android') {
  await PermissionsAndroid.requestMultiple([
    PermissionsAndroid.PERMISSIONS.RECORD_AUDIO, //For audio calls
    PermissionsAndroid.PERMISSIONS.CAMERA, // For video calls
  ]);
}
```

# Unreal Engine

Last updated : 2022-04-02 13:18:29

This document describes how to quickly integrate the TRTC SDK for Unreal Engine into your project.

## Environment Requirements

- Unreal Engine 4.27.1 or above
- **Developing for Android:**
  - Android Studio 4.0 or above
  - Visual Studio 2017 15.6 or above
  - A real device for testing
- **Developing for iOS and macOS:**
  - Xcode 11.0 or above
  - OS X 10.11 or above
  - A valid developer signature for your project
- **Developing for Windows:**
  - OS: Windows 7 SP1 or above (64-bit based on x86-64)
  - Disk space: at least 1.64 GB of space after the IDE and relevant tools are installed
  - [Visual Studio 2019](#)

## Integrating the SDK

1. Download the SDK and its [source code](#). If you have any questions, create an issue [here](#).
2. Copy the `TRTCSDK` folder to the **Source/[project\_name]** directory of your project (**[project\_name]** is the name of your project).
3. Add the following function to the **[project\_name].Build.cs** file in your project.

```
// Load the TRTC library for different platforms
private void loadTRTCSDK(ReadOnlyTargetRules Target)
{
    string _TRTCSDKPath = Path.GetFullPath(Path.Combine(ModuleDirectory, "TRTCSDK"));
    bEnableUndefinedIdentifierWarnings = false;
    if (Target.Platform == UnrealTargetPlatform.Android)
    {
```

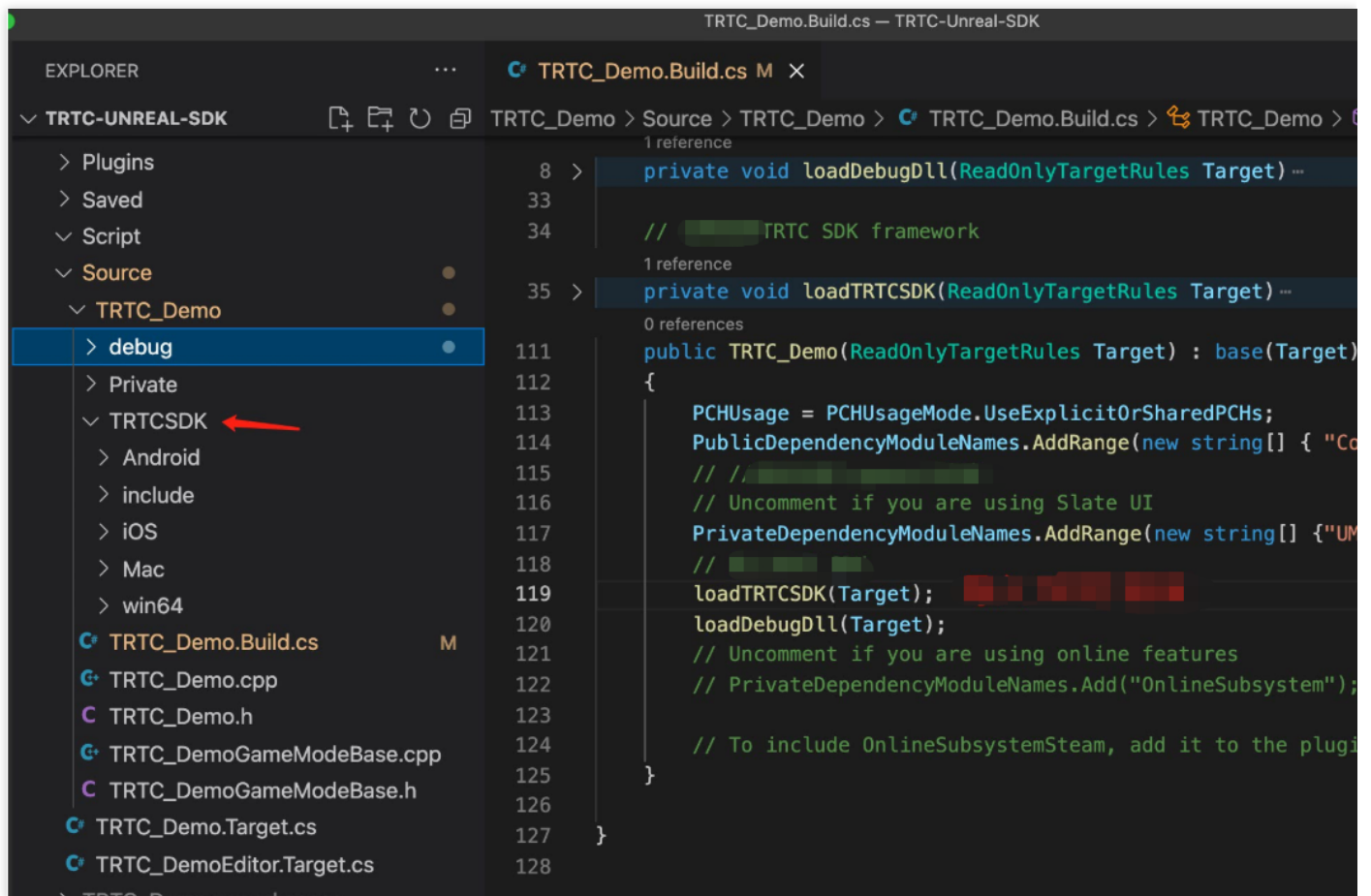
```
// Load the Android header file
PublicIncludePaths.Add(Path.Combine(_TRTCSDKPath, "include/Android"));
PrivateDependencyModuleNames.AddRange(new string[] { "Launch" });
// Load the Android APL file
AdditionalPropertiesForReceipt.Add(new ReceiptProperty("AndroidPlugin", Path.Combine(ModuleDirectory, "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)
{
// Load the iOS header file
PublicIncludePaths.Add(Path.Combine(_TRTCSDKPath, "include/iOS"));
PublicAdditionalLibraries.AddRange(new string[] {
"resolv",
"z",
"c++",
});
PublicFrameworks.AddRange(
new string[] {
"CoreML",
"VideoToolbox",
"Accelerate",
"CFNetwork",
"OpenGL",
"AVFoundation",
"CoreTelephony"
}
);
PublicAdditionalFrameworks.Add(new UEBuildFramework( "TXLiteAVSDK_TRTC", _TRTCSDKPath+"ios/TXLiteAVSDK_TRTC.framework.zip", ""));
} else if (Target.Platform == UnrealTargetPlatform.Mac)
{
// Load the macOS header file
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)
{
    // Load the 64-bit Windows header file
    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. Call the function in the **[project\_name].Build.cs** file.



5. You have integrated the TRTC SDK into your project and can now use it in the CPP file. `#include "TRTCCloud.h"`

```

// Get a TRTC singleton
#if PLATFORM_ANDROID
if (JNIEnv* Env = FAndroidApplication::GetJavaEnv()) {
void* activity = (void*) FAndroidApplication::GetGameActivityThis();
// For Android, pass in the context object
pTRTCCloud = getTRTCShareInstance(activity);
}
#else
pTRTCCloud = getTRTCShareInstance();
#endif
// Register event callbacks
pTRTCCloud->addCallback(this);
// Get the version number
std::string version = pTRTCCloud->getSDKVersion();
// Enter a room
trtc::TRTCParams params;

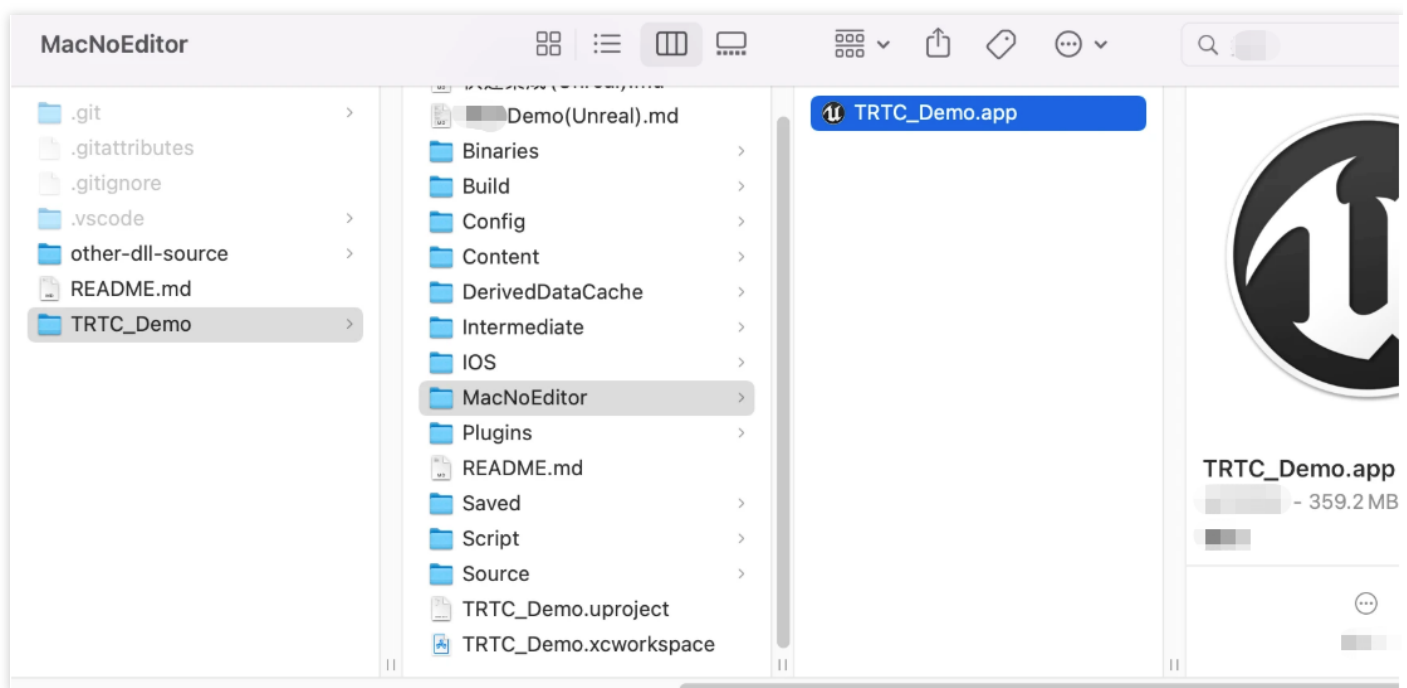
```

```
params.userId = "123";  
params.roomId = 110;  
params.sdkAppId = SDKAppID;  
params.userSig = GenerateTestUserSig().genTestUserSig(params.userId, SDKAppID, SECRETKEY);  
pTRTCCloud->enterRoom(params, trtc::TRTCApSceneVideoCall);
```

## Packaging

- macOS
- Windows
- iOS
- Android

1. Go to **File > Package Project > Mac**.
2. Configure permissions. Right-click the `xxx.app` file compiled in the previous step and select **Show Package Contents**.



3. Go to **Contents > Info.plist**.
4. Select **Information Property List** and add the following two permissions:

```
<key>NSCameraUsageDescription</key>  
<string>Video calls are possible only with camera permission.</string>  
<key>NSMicrophoneUsageDescription</key>  
<string>Audio calls are possible only with mic access.</string>
```

5. If you use UE4 Editor, add the above permissions to the **UE4Editor.app** file.

## TRTC Cross-Platform (C++) APIs

[API Documentation \(Chinese\)](#)

[API Documentation \(English\)](#)

# Demo Quick Start

## iOS and macOS

Last updated : 2022-03-30 16:15:35

This document describes how to quickly run the TRTC-API-Example for iOS and macOS.

## Environment Requirements

- Xcode 11.0 or above
- A valid developer signature for your project
- Qt Creator 4.13.3 (macOS) or above

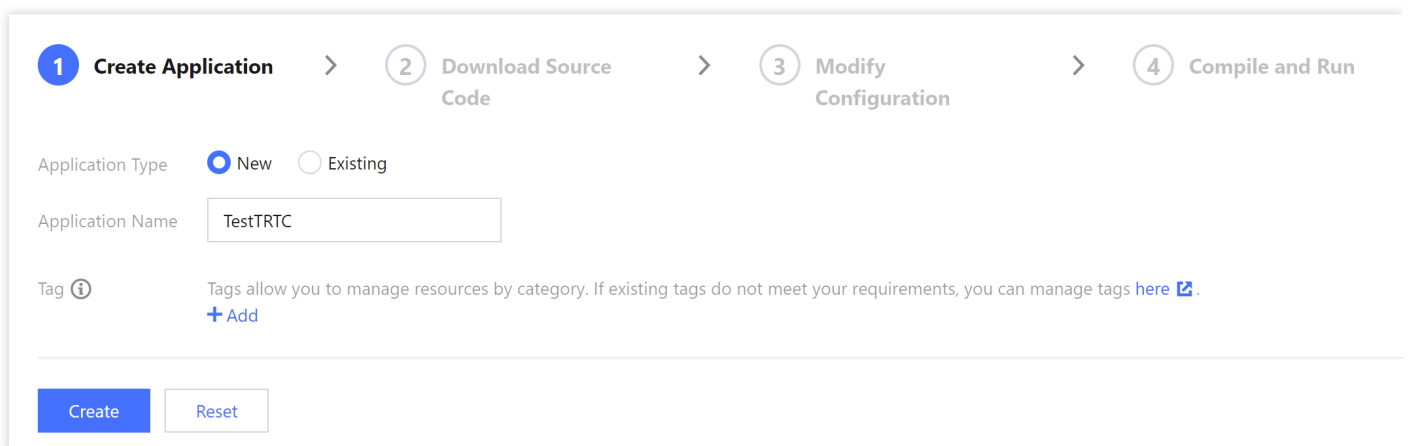
## Prerequisites

You have [signed up](#) for a Tencent Cloud account and completed [identity verification](#).

## Directions

### Step 1. Create an application

1. Log in to the TRTC console and select **Development Assistance** > [Demo Quick Run](#).
2. Select **New** and enter an application name such as `TestTRTC` . If you have already created an application, select **Existing**.
3. Add or edit tags according to your actual business needs and click **Create**.



The screenshot shows the 'Create Application' step in the TRTC console. At the top, there are four numbered steps: 1. Create Application (active), 2. Download Source Code, 3. Modify Configuration, and 4. Compile and Run. Below the steps, the 'Application Type' is set to 'New' (selected with a blue radio button) and 'Existing' is unselected. The 'Application Name' field contains 'TestTRTC'. There is a 'Tag' section with an information icon and a description: 'Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#).' Below this is a '+ Add' link. At the bottom, there are two buttons: 'Create' (blue) and 'Reset' (white).

Note :

- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to identify and organize your Tencent Cloud resources. For example, an enterprise may have multiple business units, each of which has one or more TRTC applications. In this case, the enterprise can tag TRTC applications to mark out the unit information. Tags are optional and can be added or edited according to your actual business needs.

## Step 2. Download the SDK and TRTC-API-Example source code

1. Download the SDK and TRTC-API-Example source code.
2. Click **Next**.

✓ Create Application > 2 Download Source Code > 3 Modify Configuration > 4 Compile and Run

Download SDK and Auxiliary Demo Source Code

Platform	Operation
iOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Android	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Web	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
MacOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Electron	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Windows	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Flutter	<a href="#">Download at GitHub</a>

Next

Previous

## Step 3. Configure TRTC-API-Example project files

1. In the **Modify Configuration** step, select the development platform in line with the source package downloaded.
2. Find and open `LiteAVSDK_TRTC_iOS_version number/TRTC-API-Example/Debug/GenerateTestUserSig.h`.
3. Set parameters in `GenerateTestUserSig.h` :
  - SDKAPPID: ``0`` by default. Set it to the actual ``SDKAppID``.
  - SECRETKEY: left empty by default. Set it to the actual key.

The screenshot shows the 'Modify Configuration' step in the Tencent Cloud console. The breadcrumb navigation at the top indicates the sequence: Create Application (checked) > Download Source Code (checked) > **3 Modify Configuration** > 4 Compile & Run. The main heading is 'Paste SDKAppID and Secret Key to Specified Location'. On the left, there are two input fields: 'SDKAppID' and 'Secret Key', each with a 'Copy' button. Below these fields is a warning message: '\* The secret key is sensitive information. Please do not disclose it.' At the bottom left is a blue button labeled 'Pasted and Next'. On the right, a code editor shows the 'Android' platform selected. The code is for the file 'bug/GenerateTestUserSig.java File'. It defines a class 'GenerateTestUserSig' with three static final variables: 'SDKAPPID' (set to 0), 'EXPIRETIME' (set to 604800), and 'SECRETKEY' (set to an empty string). In the original image, the values '0', '604800', and the empty string for 'SECRETKEY' are highlighted with red boxes.

4. Click **Next** to complete the creation.
5. After compilation, click **Return to Overview Page**.

Note :

- The method for generating `UserSig` described in this document involves configuring `SECRETKEY` in client code. In this method, `SECRETKEY` may be easily decompiled and reversed, and if your key is disclosed, attackers can steal your Tencent Cloud traffic. Therefore, **this method is suitable only for the local execution and debugging of TRTC-API-Example.**

- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig`. For more information, please see [How do I calculate UserSig on the server?](#).

## Step 4. Compile and run

Open the `TRTC-API-Example-OC.xcworkspace` project in the source code directory with Xcode (11.0 or above) and compile and run the `TRTC-API-Example` project.

## FAQs

### 1. Only public and private keys can be obtained when I try to view the key. How do I get a key?

TRTC SDK 6.6 (August 2019) and later versions use the new signature algorithm HMAC-SHA256. If your application was created before August 2019, you need to upgrade the signature algorithm to get a new key. Without upgrading, you can continue to use the [old algorithm ECDSA-SHA256](#). After upgrading, you can switch between the new and old algorithms as needed.

Upgrade/Switch:

1. Log in to the [TRTC console](#).
  2. Click **Application Management** in the left navigation pane, find your application, and click **Application Info**.
  3. Select the **Quick Start** tab and click **Upgrade, asymmetric encryption**, or **HMAC-SHA256** in **Step 2: obtain the secret key to issue UserSig**.
- Upgrade.



- Switch to the old algorithm ECDSA-SHA256:

**Step 2: obtain the secret key to issue UserSig**

The secret key is sensitive information. Please do not disclose it.

Secret Key (Key)

```
3d3f69f8fa29e161531ad44df1a1739d6358202539c7574c6
96e30e3caad4437
```

Copy Secret Key

\* The current mode is "HMAC-SHA256", you can switch to ["asymmetric encryption"](#).

- Switch to the new algorithm HMAC-SHA256:

**Step 2: obtain the secret key to issue UserSig**

The secret key is sensitive information. Please do not disclose it.

Public Key (PublicKey)

```
-----BEGIN PUBLIC KEY----- MFkwEwYHKoZIzj0CAQYIKo
ZlZj0DAQcDQgAE2vEazoudflwhARZDDKu8O2K0a+X eXf
WE/x1/2uXhTKN4GigjSticxrsHvntjkRAb8ohA8AUMo8qYQ
aKuB8n4w== -----END PUBLIC KEY-----
```

Copy Public Key

Private Key (PrivateKey):

```
-----BEGIN PRIVATE KEY----- MIGhAgEAMBMGBYqGSM
49AgEGCCqGSM49AwEHBG0wawIBAQQgHEvW0VaLOXi
KtJ6h 45UFV1lg6QNoWIBWj6XnG305V+hRANCAATa+MR
rOi51+XCEBFkMMq7w7YrRf5d5 d9YT/HX/a5eFMo3gaKC
NK2JzGuwe+e2OREBvyiEDwBQyjyphBoq4Hyfj -----END
PRIVATE KEY-----
```

Copy Private Key

\* The current mode is "asymmetric encryption", you can switch to ["HMAC-SHA256"](#).

## 2. The demo is running on two mobile phones, but why can't they display the images of each other?

Make sure that the two mobile phones use different UserIDs. With TRTC, you cannot use the same UserID on two devices simultaneously unless the SDKAppIDs are different.

## 3. What are firewall restrictions does the SDK face?

The SDK uses the UDP protocol for audio/video transmission and therefore cannot be used in office networks that block UDP. If you encounter such a problem, please see [How to Deal with Firewall Restrictions](#) to troubleshoot the issue.

# Android

Last updated : 2022-03-30 16:19:47

This document describes how to quickly run the TRTC-API-Example for Android.

## Environment Requirements

- Android 4.1 (SDK API level 16) or above (Android 5.0 (SDK API level 21) or above is recommended.)
- Android Studio 3.5 or above
- Devices with Android 4.1 or above

## Prerequisites

You have [signed up](#) for a Tencent Cloud account and completed [identity verification](#).

## Directions

### Step 1. Create an application

1. Log in to the TRTC console and select **Development Assistance** > [Demo Quick Run](#).
2. Select **New** and enter an application name such as `TestTRTC` . If you have already created an application, select **Existing**.
3. Add or edit tags according to your actual business needs and click **Create**.

The screenshot shows the 'Create Application' step in the TRTC console. At the top, there is a progress bar with four steps: 1. Create Application (active), 2. Download Source Code, 3. Modify Configuration, and 4. Compile and Run. Below the progress bar, the 'Application Type' is set to 'New' (selected with a blue radio button) and 'Existing' is unselected. The 'Application Name' field contains 'TestTRTC'. There is a 'Tag' section with an information icon and a description: 'Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#).' Below this is a '+ Add' link. At the bottom, there are two buttons: 'Create' (blue) and 'Reset' (white).

Note :

- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to identify and organize your Tencent Cloud resources. For example, an enterprise may have multiple business units, each of which has one or more TRTC applications. In this case, the enterprise can tag TRTC applications to mark out the unit information. Tags are optional and can be added or edited according to your actual business needs.

## Step 2. Download the SDK and TRTC-API-Example source code

1. Download the SDK and TRTC-API-Example source code.
2. Click **Next**.

✓ Create Application > **2 Download Source Code** > ③ Modify Configuration > ④ Compile and Run

 Download SDK and Auxiliary Demo Source Code

Platform	Operation
iOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Android	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Web	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
MacOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Electron	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Windows	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Flutter	<a href="#">Download at GitHub</a>

Next

Previous

## Step 3. Configure TRTC-API-Example project files

1. In the **Modify Configuration** step, select the development platform in line with the source package downloaded.

- Find and open `LiteAVSDK_TRTC_Android` version number/TRTC-API-Example/Debug/src/main/java/com/tencent/trtc/debug/GenerateTestUserSig.java .
- Set parameters in `GenerateTestUserSig.java` as follows:
  - SDKAPPID: a placeholder by default. Set it to the actual `SDKAppID`.
  - `SECRETKEY`: a placeholder by default. Set it to the actual key.

✓ Create Application

✓ Download Source Code

3 Modify Configuration

4 Compile & Run

### Paste SDKAppID and Secret Key to Specified Location

SDKAppID

Copy

Secret Key

Copy

\* The secret key is sensitive information. Please do not disclose it.

Pasted and Next

Decompress the source package downloaded in Step 2, and open Android/TRTCScenesDemo/debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java File

Android iOS&macOS Windows(C++) Windows(C#) Web Mini Program Electron

```
public class GenerateTestUserSig {  
  
    private static final int SDKAPPID = 0;  
  
    private static final int EXPIRETIME = 604800;  
  
    private static final String SECRETKEY = "";
```

- Click **Next** to complete the creation.
- After compilation, click **Return to Overview Page**.

#### Note :

- The method for generating `UserSig` described in this document involves configuring `SECRETKEY` in client code. In this method, `SECRETKEY` may be easily decompiled and reversed, and if your key is disclosed, attackers can steal your Tencent Cloud traffic. Therefore, **this method is suitable only for the local execution and debugging of TRTC-API-Example.**
- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig` . For more information, please see [How do I calculate UserSig on the server?](#).

## Step 4. Compile and run

Open the `TRTC-API-Example` project with Android Studio (3.5 or above) and click **Run**.

## FAQs

### 1. Only public and private keys can be obtained when I try to view the key. How do I get a key?

TRTC SDK 6.6 (August 2019) and later versions use the new signature algorithm HMAC-SHA256. If your application was created before August 2019, you need to upgrade the signature algorithm to get a new key. Without upgrading, you can continue to use the [old algorithm ECDSA-SHA256](#). After upgrading, you can switch between the new and old algorithms as needed.

Upgrade/Switch:

1. Log in to the [TRTC console](#).
  2. Click **Application Management** in the left navigation pane, find your application, and click **Application Info**.
  3. Select the **Quick Start** tab and click **Upgrade, asymmetric encryption**, or **HMAC-SHA256** in **Step 2: obtain the secret key to issue UserSig**.
- Upgrade.
  - Switch to the old algorithm ECDSA-SHA256:

**Step 2: obtain the secret key to issue UserSig**

The secret key is sensitive information. Please do not disclose it.

Secret Key (Key)

0f040c3cce90c302af03b3851c90b178d5e38ff39459b9  
bccc87c113a72b5aa

Copy Secret Key

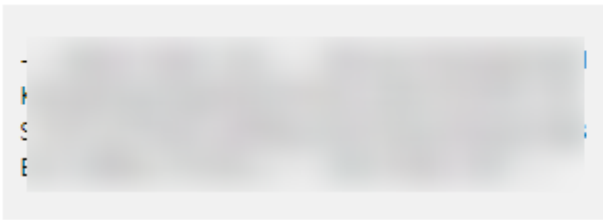
\* The current mode is "HMAC-SHA256", you can switch to ["asymmetric encryption"](#).

- Switch to the new algorithm HMAC-SHA256:

**Step 2: obtain the secret key to issue UserSig**

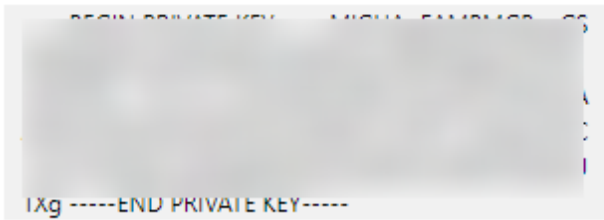
The secret key is sensitive information. Please do not disclose it.

Public Key (PublicKey)



Copy Public Key

Private Key (PrivateKey):



Copy Private Key

\* The current mode is "asymmetric encryption", you can switch to **HMAC-SHA256**.

## 2. The app is running on two mobile phones, but why can't they display the images of each other?

Make sure that the two mobile phones use different UserIDs. With TRTC, you cannot use the same UserID on two devices simultaneously unless the SDKAppIDs are different.

## 3. What are firewall restrictions does the SDK face?

The SDK uses the UDP protocol for audio/video transmission and therefore cannot be used in office networks that block UDP. If you encounter such a problem, please see [How to Deal with Firewall Restrictions](#) to troubleshoot the issue.

# Windows

Last updated : 2022-03-30 16:23:48

This document describes how to quickly run the TRTC demo for Windows.

## Environment Requirements

### Developing for Windows using C++

- Microsoft Visual Studio (VS) 2015 or above (2015 is recommended)
- TRTC SDK for Windows 8.0 or above (8.1 is recommended)

### Developing for Windows using C#

- VS 2015 or above (2017 is recommended)
- .Net Framework 4.0 or above (4.0 is recommended)

### Developing for Windows using Qt

- VS 2015 or above (2015 is recommended)
- Find the right version of Qt add-in for your VS on the official website, and download and install the [VSIX](#) file.
- Open VS, in the menu bar, select `QT VS Tools > Qt Options > Qt Versions` , and add an MSVC compiler.
- Copy all the DLL files in `SDK/CPlusPlus/Win32/lib` to the `debug / release` folder of the project directory.

Note :

The `debug / release` folder is created automatically after environment setup is completed in VS.

## Prerequisites

You have [signed up](#) for a Tencent Cloud account and completed [identity verification](#).

# Directions

## Step 1. Create an application

1. Log in to the TRTC console and select **Development Assistance** > **Demo Quick Run**.
2. Select **New** and enter an application name such as `TestTRTC` . If you have already created an application, select **Existing**.
3. Add or edit tags according to your actual business needs and click **Create**.

1 Create Application > 2 Download Source Code > 3 Modify Configuration > 4 Compile and Run

Application Type ☒ New ☐ Existing

Application Name

Tag ⓘ Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#) .

[+ Add](#)

### Note :

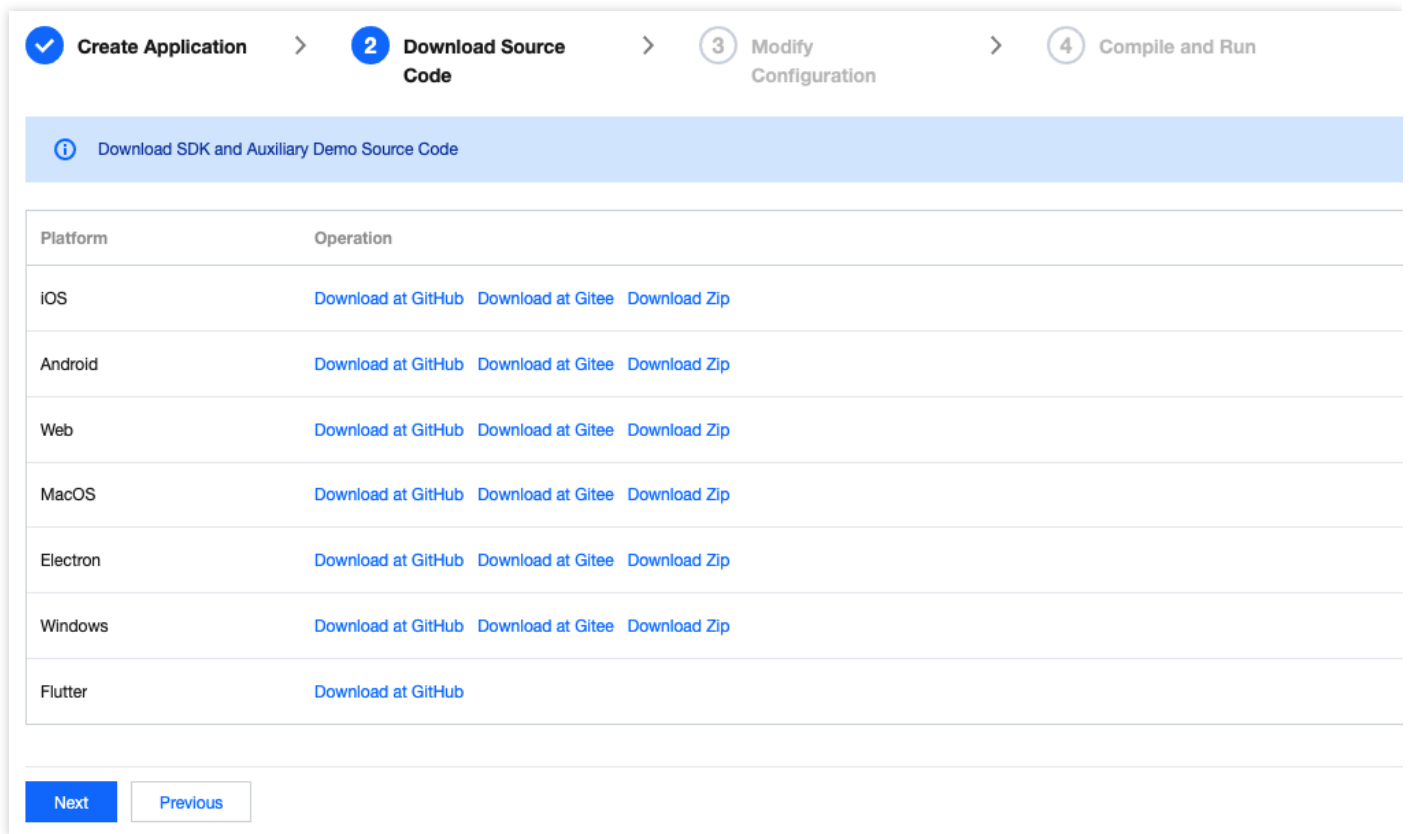
- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to identify and organize your Tencent Cloud resources. For example, an enterprise may have multiple business units, each of which has one or more TRTC applications. In this case, the enterprise can tag TRTC applications to mark out the unit information. Tags are optional and can be added or edited according to your actual business needs.

## Step 2. Download the SDK and demo source code

1. Download the SDK and demo source code for your platform.



## 2. Click **Next**.



## Step 3. Configure demo project files

1. In the **Modify Configuration** step, select the development platform in line with the source package downloaded.
2. Find and open the `GenerateTestUserSig` file:

Platform	Relative Path to File
Windows (C++)	Windows/DuilibDemo/GenerateTestUserSig.h
Windows (C#)	Windows/CSharpDemo/GenerateTestUserSig.cs

3. Set parameters in the `GenerateTestUserSig.js` file:
  - SDKAPPID: ``0`` by default. Set it to the actual ``SDKAppID``.
  - SECRETKEY: left empty by default. Set it to the actual key.

✓ Create Application

✓ Download Source Code

3 Modify Configuration

4 Compile & Run

### Paste SDKAppID and Secret Key to Specified Location

SDKAppID

Copy

Secret Key

Copy

\* The secret key is sensitive information. Please do not disclose it.

Pasted and Next

Decompress the source package downloaded in Step 2, and open Android/TRTCSdemosDemo/debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java File

Android iOS&macOS Windows(C++) Windows(C#) Web Mini Program Electron

```

public class GenerateTestUserSig {

    private static final int SDKAPPID = 0;

    private static final int EXPIRETIME = 604800;

    private static final String SECRETKEY = "";

```

4. Click **Next** to complete the creation.

5. After compilation, click **Return to Overview Page**.

Note :

- The method for generating `UserSig` described in this document involves configuring `SECRETKEY` in client code. In this method, `SECRETKEY` may be easily decompiled and reversed, and if your key is leaked, attackers can steal your Tencent Cloud traffic. Therefore, **this method is only suitable for the local execution and debugging of the demo.**
- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig`. For more information, please see [How do I calculate UserSig on the server?](#).

## Step 4. Compile and run

### • Windows (C++):

Use VS (VS 2015 is recommended) to open `DuiLibDemo\TRTCDuiLibDemo.sln` in the source code directory, and compile and run the demo project. We recommend building the project in the release mode for x86.

### • Windows (C#):

Use VS (VS 2017 is recommended) to open `CSharpDemo\TRTCCSharpDemo.sln` in the source code

directory, and compile and run the demo project. We recommend building the project in the release mode for x86.

- **Windows (QT):**

Use VS (VS 2015 is recommended) to open

`QTDemo\QTDemo.pro` in the source code directory, and compile and run the QTDemo project.

## FAQs

### 1. Only public and private keys can be obtained when I try to view the key. How do I get a key?

TRTC SDK 6.6 (August 2019) and later versions use the new signature algorithm HMAC-SHA256. If your application was created before August 2019, you need to upgrade the signature algorithm to get a new key. Without upgrading, you can continue to use the [old algorithm ECDSA-SHA256](#). After upgrading, you can switch between the new and old algorithms as needed.

Upgrade/Switch:

1. Log in to the [TRTC console](#).
  2. Click **Application Management** in the left navigation pane, find your application, and click **Application Info**.
  3. Select the **Quick Start** tab and click **Upgrade, asymmetric encryption**, or **HMAC-SHA256** in **Step 2: obtain the secret key to issue UserSig**.
- Upgrade.
  - Switch to the old algorithm ECDSA-SHA256:

#### Step 2: obtain the secret key to issue UserSig

The secret key is sensitive information. Please do not disclose it.

Secret Key (Key)

```
3d3f69f8fa29e161531ad44df1a1739d6358202539c7574c6
96e30e3caad4437
```

Copy Secret Key

\* The current mode is "HMAC-SHA256", you can switch to ["asymmetric encryption"](#).

- Switch to the new algorithm HMAC-SHA256:

**Step 2: obtain the secret key to issue UserSig**

The secret key is sensitive information. Please do not disclose it.

Public Key (PublicKey)

```
-----BEGIN PUBLIC KEY----- MFkwEwYHKoZIzj0CAQYIKo
ZlZj0DAQcDQgAE2vjEazoudflwhARZDDKu8O2K0a+X eXf
WE/x1/2uXhTKN4GigjSticxrsHvntjkRAb8ohA8AUMo8qYQ
aKuB8n4w== -----END PUBLIC KEY-----
```

Copy Public Key

Private Key (PrivateKey):

```
-----BEGIN PRIVATE KEY----- MIGHAgEAMBMGBYqGSM
49AgEGCCqGSM49AwEHBG0wawIBAQQgHEvW0VaLOXi
KtJ6h 45UfFv1lg6QNoWIBWj6XnG305V+hRANCAATa+MR
rOi51+XCEBFkMMq7w7YrRr5d5 d9YT/HX/a5eFMo3gaKC
NK2JzGuwe+e2OREBvyiEDwBQjyphBoq4Hyfj -----END
PRIVATE KEY-----
```

Copy Private Key

\* The current mode is "asymmetric encryption", you can switch to **"HMAC-SHA256"**.

## 2. The demo is running on two devices, but why can't they display the images of each other?

Make sure that the two devices use different `UserIDs` . With TRTC, you cannot use the same `UserID` on two devices simultaneously unless the `SDKAppIDs` are different.

## 3. What are firewall restrictions does the SDK face?

The SDK uses the UDP protocol for audio/video transmission and therefore cannot be used in office networks that block UDP. If you encounter such a problem, please see [How to Deal with Firewall Restrictions](#).

# Web

Last updated : 2022-04-15 15:49:59

This document describes how to quickly run the demo for the TRTC web SDK.

## Preparations

Before you run the demo for the TRTC web SDK, pay attention to the following:

### Supported platforms

The TRTC web SDK is based on WebRTC. For details about the browsers supported, please see [Supported Platforms](#).

If your browser (for example, WebView) is not in the list, you can run a [TRTC Web SDK Support Level Test](#) in the browser to test whether it fully supports WebRTC.

If your application scenario is mainly in the education sector, consider using the [TRTC Electron SDK](#), which supports the dual-stream mode (big and small images), with more flexible screen sharing schemes and better recovery capabilities for poor network connections.

### URL protocol support

Because of the security policies of browsers, when you use WebRTC, there are requirements on the protocol used for access. For details, see the table below.

Scenario	Protocol	Receive (Playback)	Send (Publish)	Share Screen	Remarks
Production	HTTPS	Supported	Supported	Supported	<b>Recommended</b>
Production	HTTP	Supported	Not supported	Not supported	
Local development	http://localhost	Supported	Supported	Supported	<b>Recommended</b>
Local development	http://127.0.0.1	Supported	Supported	Supported	
Local development	http://[local IP address]	Supported	Not supported	Not supported	

Scenario	Protocol	Receive (Playback)	Send (Publish)	Share Screen	Remarks
Local development	file:///	Supported	Supported	Supported	

## Firewall configuration

The TRTC web SDK uses the following ports and domain names for data transfer, which should be added to the allowlist of your firewall. You can use our [demo](#) to check whether the configuration has taken effect. For details, see [Dealing with Firewall Restrictions](#).

- TCP port: 8687
- UDP ports: 8000, 8080, 8800, 843, 443, 16285
- Domain names: \*.rtc.qq.com , yun.tim.qq.com

## Prerequisites

You have [signed up](#) for a Tencent Cloud account and completed [identity verification](#).

## Directions


### Step 1. Create an application

1. In the TRTC console, click **Development Assistance** > [Demo Quick Run](#).
2. Select **New** and enter an application name such as `TestTRTC` . If you have already created an application, select **Existing**.
3. Add or edit tags according to your actual business needs and click **Create**.

1 Create Application > 2 Download Source Code > 3 Modify Configuration > 4 Compile and Run

Application Type ☒ New ☐ Existing

Application Name

Tag ⓘ Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#) .  
[+ Add](#)

Create

Reset

Note :

- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to mark and sort your Tencent Cloud resources. For example, your company may have multiple business units, each using one or more TRTC applications. You can tag the applications with the information of the corresponding business units. Tags are optional. You can add and edit tags for your applications as needed.

## Step 2. Download the SDK and demo source code

1. Download the web SDK and demo source code.
2. Click **Next**.

- ✓ Create Application > 2 Download Source Code > 3 Modify Configuration > 4 Compile and Run

i Download SDK and Auxiliary Demo Source Code

Platform	Operation
iOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Android	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Web	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
MacOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Electron	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Windows	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Flutter	<a href="#">Download at GitHub</a>

Next

Previous

### Step 3. Get the SDKAppID and secret key

1. In the **Modify Configuration** step, note the `SDKAppID` and secret key.
2. Paste the `SDKAppID` and secret key and click **Next**.



- ✓ Create Application > ✓ Download Source Code > **3 Modify Configuration** > 4 Compile & Run

### Paste SDKAppID and Secret Key to Specified Location

SDKAppID

Secret Key

\* The secret key is sensitive information. Please do not disclose it.

Pasted and Next

Decompress the source package downloaded in Step 2, and open Android/TRTCSdemosDemo/debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java File

**Android** iOS&macOS Windows(C++) Windows(C#) Web Mini Program Electron

```
public class GenerateTestUserSig {  
  
    private static final int SDKAPPID = 0;  
  
    private static final int EXPIRETIME = 604800;  
  
    private static final String SECRETKEY = "";
```

## Step 4. Run the demo

We offer the following demos for the TRTC web SDK to meet different customer needs:

- **base-js** : The TRTC web standard demo, which integrates capabilities including audio/video calls and device selection. It was developed using jQuery and can be run in a browser. You can try it out [here](#).
  - **quick-demo-js** : The TRTC web quick demo (JavaScript), which integrates capabilities including audio/video call and device selection. It was developed based on JavaScript without using any libraries and can be run in a browser. You can try it out [here](#).
  - **quick-demo-vue2-js** : The TRTC web quick demo (Vue.js 2.0), which integrates capabilities including audio/video call and device selection. It was developed using Vue.js 2.0. To use it, you need to install Node.js. You can try it out [here](#).
- Demo 1: base-js
  - Demo 2: quick-demo-js
  - Demo 3: quick-demo-vue2-js

1. Find and open `TRTC_Web/base-js/js/debug/GenerateTestUserSig.js` .
2. Set parameters in the `GenerateTestUserSig.js` file as follows:

- SDKAPPID: 0 by default. Set it to the actual `SDKAppID` .

- `SECRETKEY` : Left empty by default. Set it to the actual key.

### 3. Run the demo:

Open `index.html` in the root directory of the demo with Chrome to run the demo.

#### Note

- Normally, the demo needs to be deployed on the server and then accessed through `https://domain name/xxx`. You can also build a server locally and access the demo through `localhost:port`.
- Currently, the desktop version of Chrome offers better support for the features of the TRTC web SDK; therefore, Chrome is recommended.

- Click **Join** to join a room and publish the local stream.

You can open multiple pages and click **Join Room** on each of them. You should be able to see multiple videos, which simulate a real-time audio/video call.

- Click **Camera Select** to select a camera.
- Click **Microphone Select** to select a mic.

#### Note

WebRTC uses the camera and mic of your device to capture audio and video. During the demo run, when prompted by Chrome, you should click **Allow**.

#### Note :

- The method used to generate `UserSig` in this document is to configure the secret key in the client code. This makes the key vulnerable to decompilation and reverse engineering. Once your key is leaked, attackers can steal your Tencent Cloud traffic. Therefore, **this method is only suitable for locally running a demo project and debugging**.
- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig`. For more information, see [How do I calculate UserSig on the server?](#).

## FAQs

### 1. There is only information of the public and private keys when I try to view the secret key. How do I get the secret key?

Since the release of v6.6 for app and v4.0 for web in August 2019, the new signature algorithm HMAC-SHA256 has been used. If your application was created before August 2019, you need to upgrade the signature algorithm to get a new key. Without upgrading, you can continue to use the [old algorithm ECDSA-SHA256](#). After upgrading, you can switch between the new and old algorithms as needed.

Upgrade/Switch:

1. Log in to the [TRTC console](#).
  2. Click **Application Management** on the left sidebar, find your application, and click **Application Info**.
  3. Select the **Quick Start** tab and click **Upgrade, asymmetric encryption**, or **HMAC-SHA256** in **Step 2: obtain the secret key to issue UserSig**.
- Upgrade
  - Switch to the old algorithm ECDSA-SHA256:

#### Step 2: obtain the secret key to issue UserSig

The secret key is sensitive information. Please do not disclose it.

Secret Key (Key)

```
3d3f69f8fa29e161531ad44df1a1739d6358202539c7574c6
96e30e3caad4437
```

Copy Secret Key

\* The current mode is "HMAC-SHA256", you can switch to "[asymmetric encryption](#)".

- Switch to the new algorithm HMAC-SHA256:

**Step 2: obtain the secret key to issue UserSig**

The secret key is sensitive information. Please do not disclose it.

Public Key (PublicKey)

```
-----BEGIN PUBLIC KEY----- MFkwEwYHKoZIzj0CAQYIKo
ZlZj0DAQcDQgAE2vjEazoudflwhARZDDKu8O2K0a+X eXf
WE/x1/2uXhTKN4GigjSticxrsHvntjkRab8ohA8AUMo8qYQ
aKuB8n4w== -----END PUBLIC KEY-----
```

Copy Public Key

Private Key (PrivateKey):

```
-----BEGIN PRIVATE KEY----- MIGHAQEAMBMGBYqGSM
49AgEGCCqGSM49AwEHBG0wawIBAQQgHEvW0VaLOXi
KtJ6h 45UfFv1lg6QNoWIBWj6XnG305V+hRANCAATa+MR
rOi51+XCEBFkMMq7w7YrRr5d5 d9YT/HX/a5eFMo3gaKC
NK2JzGuwe+e2OREBvyiEDwBQylyphBoq4Hyfj -----END
PRIVATE KEY-----
```

Copy Private Key

\* The current mode is "asymmetric encryption", you can switch to **HMAC-SHA256**.

## 2. What should I do if the client error "RtcError: no valid ice candidate found" occurs?

This error indicates that the TRTC web SDK failed with regard to hole punching via Session Traversal Utilities for NAT (STUN). Please check your firewall configuration against the [Environment Requirements](#).

## 3. What should I do if the client error "RtcError: ICE/DTLS Transport connection failed" or "RtcError: DTLS Transport connection timeout" occurs?

It indicates that the TRTC web SDK failed to establish a media transmission channel. Please check your firewall configuration against the [Environment Requirements](#).

## 4. What should I do if a 10006 error occurs?

If the error "Join room failed result: 10006 error: service is suspended, if charge is overdue, renew it" occurs, check whether the TRTC service status for your application is "normal".

Log in to the [TRTC console](#), select the application you created, and click **Application Info** to view its service status.

# TRTC Service Status

Status **Available**

Note :

For other questions, see [Web](#).

# Electron

Last updated : 2022-03-30 16:51:29

This document describes how to quickly run the TRTC demo for Electron.

## Prerequisites

You have [signed up](#) for a Tencent Cloud account and completed [identity verification](#).

## Directions

### Step 1. Create an application

1. Log in to the TRTC console and select **Development Assistance** > **Demo Quick Run**.
2. Select **New** and enter an application name such as `TestTRTC` . If you have already created an application, select **Existing**.
3. Add or edit tags according to your actual business needs and click **Create**.

The screenshot shows the 'Create Application' step in the TRTC console. At the top, there is a progress bar with four steps: 1. Create Application (active), 2. Download Source Code, 3. Modify Configuration, and 4. Compile and Run. Below the progress bar, the 'Application Type' is set to 'New' (selected with a blue radio button) and 'Existing' is unselected. The 'Application Name' field contains 'TestTRTC'. Below this, there is a 'Tag' section with an information icon and a text description: 'Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#).' There is a '+ Add' link below the text. At the bottom, there are two buttons: 'Create' (blue) and 'Reset' (white).

#### Note :

- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to identify and organize your Tencent Cloud resources. For example, an enterprise may have multiple business units, each of which has one or more TRTC applications. In this case, the enterprise can tag TRTC applications to mark out the unit

information. Tags are optional and can be added or edited according to your actual business needs.

## Step 2. Download the SDK and demo source code

1. Download the SDK and demo source code for your platform.
2. Click **Next**.

Platform	Operation
iOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Android	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Web	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
MacOS	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Electron	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Windows	<a href="#">Download at GitHub</a> <a href="#">Download at Gitee</a> <a href="#">Download Zip</a>
Flutter	<a href="#">Download at GitHub</a>

## Step 3. Configure demo project files

1. In the **Modify Configuration** step, select the development platform in line with the source package downloaded.
2. Find and open the `Electron/js/GenerateTestUserSig.js` file.
3. Set parameters in the `GenerateTestUserSig.js` file:
  - SDKAPPID: ``0`` by default. Set it to the actual ``SDKAppID``.
  - SECRETKEY: left empty by default. Set it to the actual key.

✓ Create Application

✓ Download Source Code

3 Modify Configuration

4 Compile & Run

### Paste SDKAppID and Secret Key to Specified Location

SDKAppID

Copy

Secret Key

Copy

\* The secret key is sensitive information. Please do not disclose it.

Pasted and Next

Decompress the source package downloaded in Step 2, and open Android/TRTCScenesDemo/debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java File

Android

iOS&macOS

Windows(C++)

Windows(C#)

Web

Mini Program

Electron

```

public class GenerateTestUserSig {

    private static final int SDKAPPID = 0;

    private static final int EXPIRETIME = 604800;

    private static final String SECRETKEY = "";

```

4. Click **Next** to complete the creation.

5. After compilation, click **Return to Overview Page**.

Note :

- The method for generating `UserSig` described in this document involves configuring `SECRETKEY` in client code. In this method, `SECRETKEY` may be easily decompiled and reversed, and if your key is leaked, attackers can steal your Tencent Cloud traffic. Therefore, **this method is only suitable for the local execution and debugging of the demo.**
- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig`. For more information, see [How do I calculate UserSig on the server?](#).

### File paths and description:

```

.
|---README.md README file. Please read it carefully.
|---main.electron.js Main Electron file
|---public Static files
|---babel.config.js
|---package.json

```



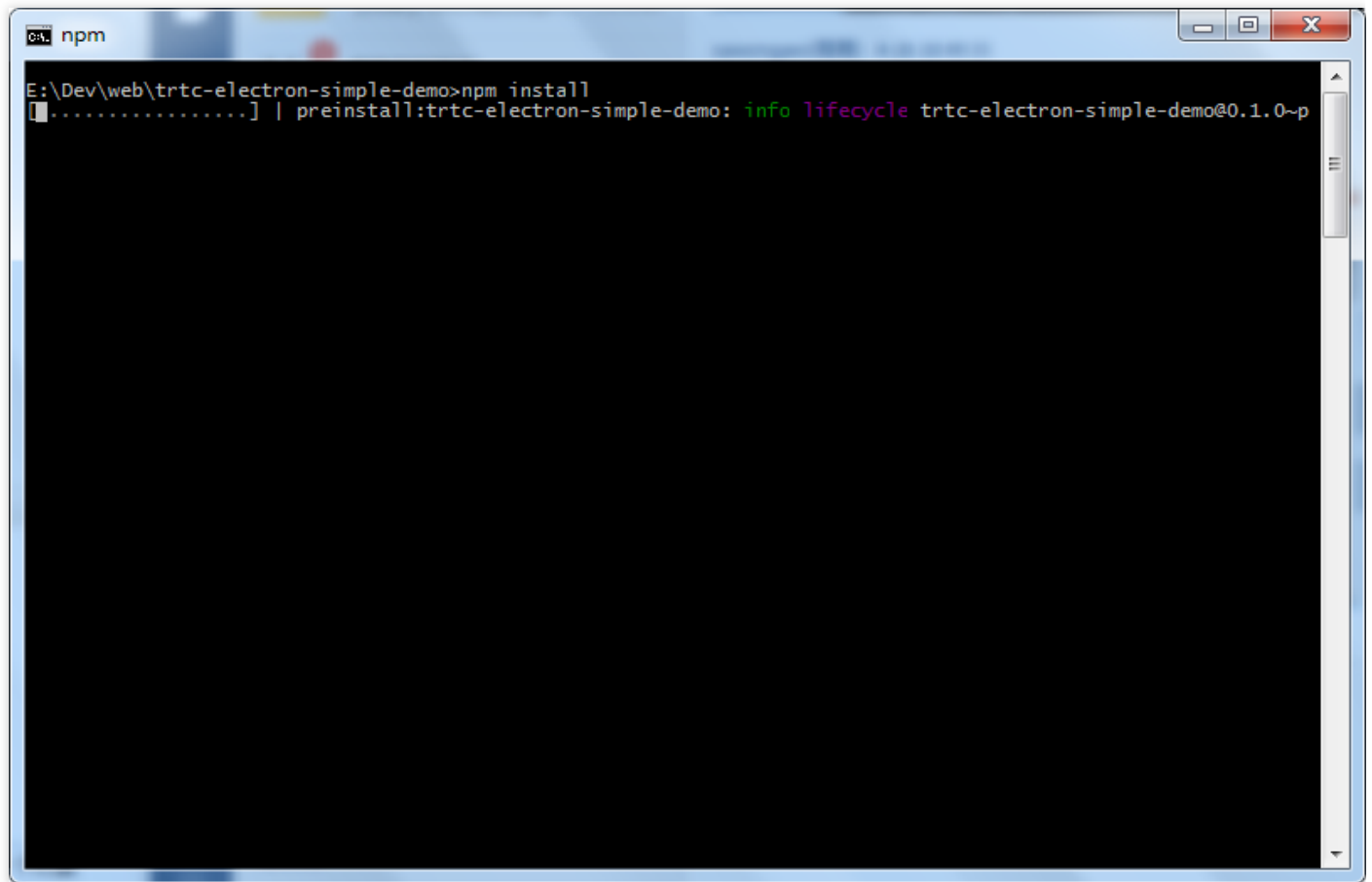
```
|---vue.config.js Vue CLI project file
|---src Source code directory
| |---app.vue
| |---common.css
| |---main.js
| |---components UI component
| | |---main-menu.vue
| | |---nav-bar.vue
| | |---show-screen-capture.vue
| |---common Utility functions, public libraries, etc.
| | |---live-room-service.js
| | |---log.js Log tools
| | |---mtah5.js
| | |---routes.js
| | |---rand.js
| |---pages Views
| | |---index.vue Homepage
| | |---trtc Video conferencing views
| | | |---trtc-room.vue Video conferencing room view
| | | |---trtc-index.vue Video conferencing entry view
| | |---404.vue
| | |---live Live streaming views
| | | |---live-index.vue Live streaming entry view
| | | |---live-room-audience.vue Audience room view
| | | |---live-room-anchor.vue Anchor room view
| |---debug When deploying your project, please move the signature logic in this folder to the server for implementation.
| | |---lib-generate-test-usersig.min.js
| | |---gen-test-user-sig.js
```

## Step 4. Compile and run

- Windows
- macOS

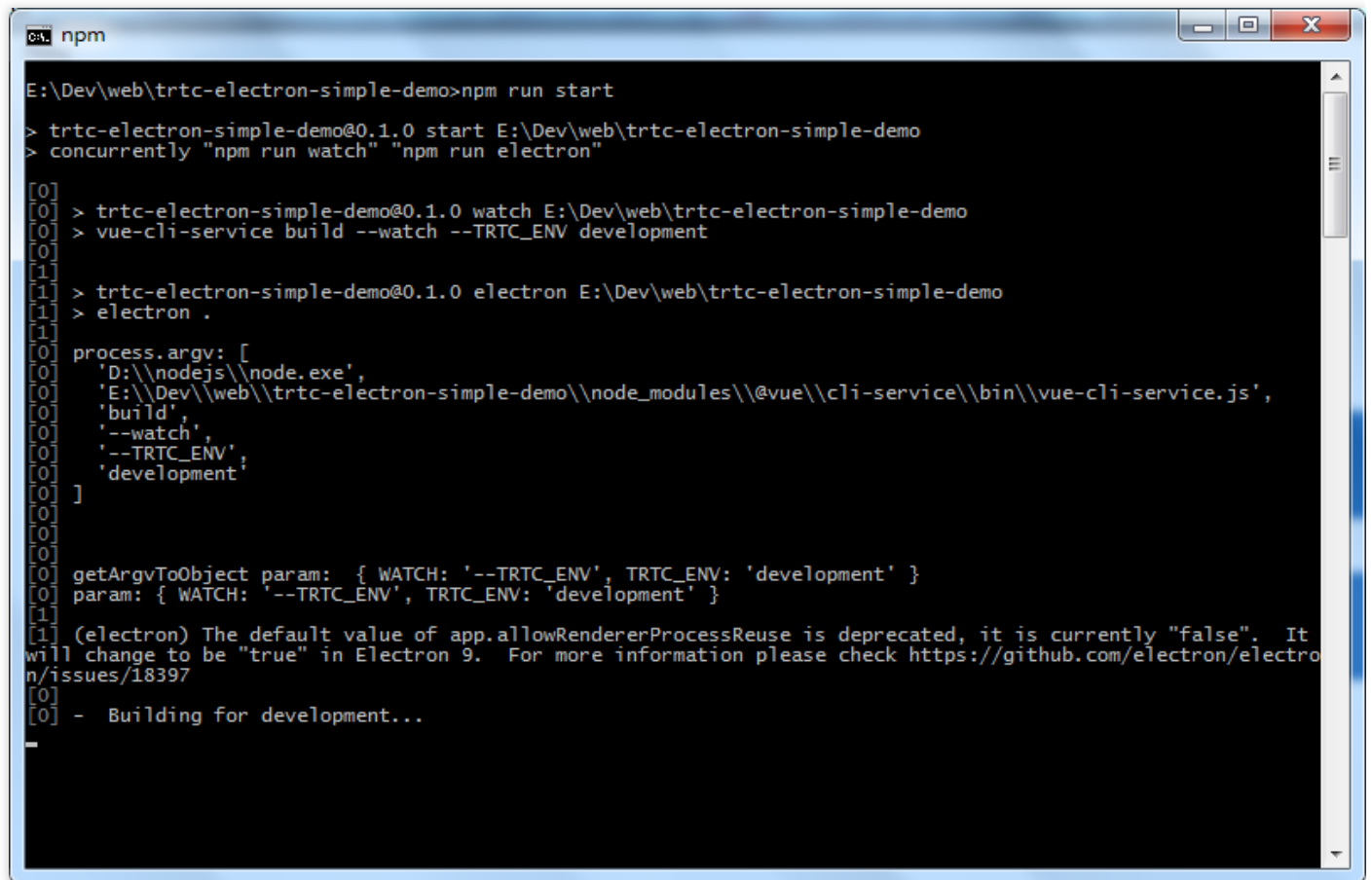
1. Install the latest version of Node.js. We recommend you use a 64-bit MSI file for the installation. Download [here](#).
2. Press Windows+R and type `cmd` to open the Command Prompt as an administrator, locate the [project directory](#), and run the following command.

```
$ npm install
```



3. After the npm dependencies are installed, run the following command in the Command Prompt to run the demo.

```
$ npm run start # On first run, it may take a while to show the UI.
```



```
ca. npm
E:\Dev\web\trtc-electron-simple-demo>npm run start
> trtc-electron-simple-demo@0.1.0 start E:\Dev\web\trtc-electron-simple-demo
> concurrently "npm run watch" "npm run electron"

[0] > trtc-electron-simple-demo@0.1.0 watch E:\Dev\web\trtc-electron-simple-demo
[0] > vue-cli-service build --watch --TRTC_ENV development

[1] > trtc-electron-simple-demo@0.1.0 electron E:\Dev\web\trtc-electron-simple-demo
[1] > electron .

process.argv: [
  'D:\\nodejs\\node.exe',
  'E:\\Dev\\web\\trtc-electron-simple-demo\\node_modules\\@vue\\cli-service\\bin\\vue-cli-service.js',
  'build',
  '--watch',
  '--TRTC_ENV',
  'development'
]

getArgvToObject param: { WATCH: '--TRTC_ENV', TRTC_ENV: 'development' }
param: { WATCH: '--TRTC_ENV', TRTC_ENV: 'development' }

[1] (electron) The default value of app.allowRendererProcessReuse is deprecated, it is currently "false". It
will change to be "true" in Electron 9. For more information please check https://github.com/electron/electro
n/issues/18397
[0] - Building for development...
```

## Main project commands

Command	Description
npm run start	Runs the demo in development environment.
npm run pack:mac	Packages the project into a DMG installer for macOS.
npm run pack:win64	Packages the project into a 64-bit EXE installer for Windows.

## FAQs

### 1. Only public and private keys can be obtained when I try to view the key. How do I get a key?

TRTC SDK 6.6 (August 2019) and later versions use the new signature algorithm HMAC-SHA256. If your application was created before August 2019, you need to upgrade the signature algorithm to get a new key. Without upgrading, you can continue to use the [old algorithm ECDSA-SHA256](#).

**Upgrade:**

1. Log in to the [TRTC console](#).
2. Click **Application Management** on the left sidebar, find your application, and click **Application Info**.
3. Select the **Quick Start** tab and click **upgrade** in **Step 2: obtain the secret key to issue UserSig**.

**2. The demo is running on two devices, but why can't they display the images of each other?**

Make sure that the two devices use different `UserIDs` . With TRTC, you cannot use the same `UserID` on two devices simultaneously unless the `SDKAppIDs` are different.

**3. What firewall restrictions does the SDK face?**

The SDK uses the UDP protocol for audio/video transmission and therefore cannot be used in office networks that block UDP. If you encounter such a problem, see [How to Deal with Firewall Restrictions](#).

Note :

For more FAQs, see [Electron](#).

## Technical Support

[Contact us](#) if you have any questions.

## References

- [SDK API Guide](#)
- [SDK Update Log](#)
- [Simple Demo Source Code](#)
- [API Example Source Code](#)
- [FAQs](#)

# Flutter

Last updated : 2022-03-30 16:59:35

This document describes how to quickly run the TRTC demo for Flutter.

Note :

At present, the Flutter SDK only supports Android and iOS

## Environment Requirements

- Flutter 2.0 or above
- **Developing for Android:**
  - Android Studio 3.5 or above
  - Devices with Android 4.1 or above
- **Developing for iOS and macOS:**
  - Xcode 11.0 or above
  - OS X 10.11 or above
  - A valid developer signature for your project
- **Developing for Windows:**
  - OS: Windows 7 SP1 or above (64-bit based on x86-64)
  - Disk space: at least 1.64 GB of space after the IDE and relevant tools are installed
  - Install [Visual Studio 2019](#)

## Prerequisites

You have [signed up](#) for a Tencent Cloud account and completed identity verification.

## Directions

### Step 1. Create an application

1. Log in to the TRTC console and select **Development Assistance** > [Demo Quick Run](#).
2. Select **New** and enter an application name such as `TestTRTC` . If you have already created an application, select **Existing**.

3. Add or edit tags according to your actual business needs and click **Create**.

The screenshot shows the 'Create Application' step in the Tencent Cloud console. At the top, there is a progress bar with four steps: 1. Create Application (active), 2. Download Source Code, 3. Modify Configuration, and 4. Compile and Run. Below the progress bar, the 'Application Type' is set to 'New' (selected with a radio button) and 'Existing' is unselected. The 'Application Name' field contains 'TestTRTC'. There is a 'Tag' section with an information icon and a description: 'Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#).' Below this is a '+ Add' link. At the bottom, there are two buttons: 'Create' (blue) and 'Reset' (white).

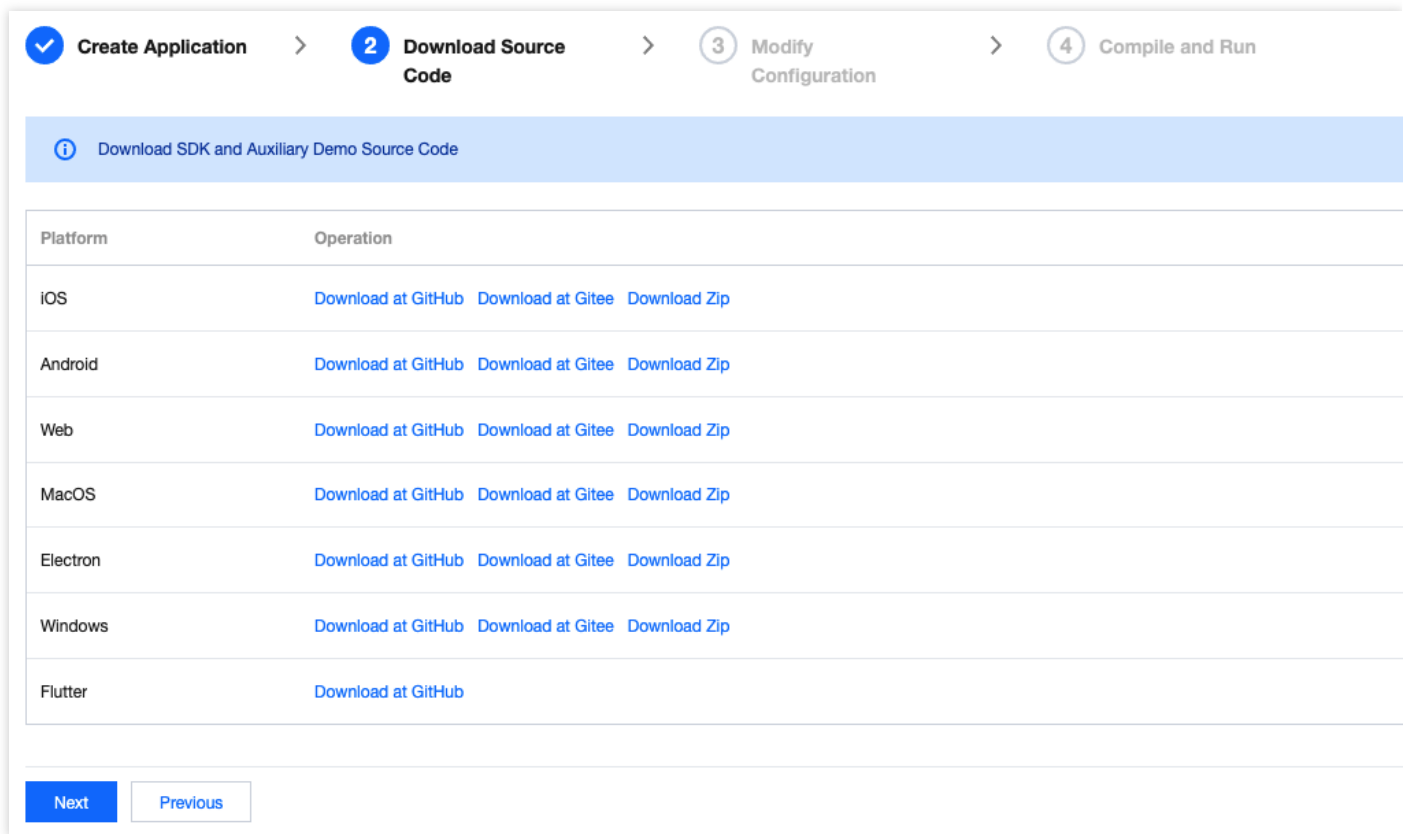
Note :

- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to identify and organize your Tencent Cloud resources. For example, an enterprise may have multiple business units, each of which has one or more TRTC applications. In this case, the enterprise can tag TRTC applications to mark out the unit information. Tags are optional and can be added or edited according to your actual business needs.

## Step 2. Download the SDK and demo source code

1. Download the SDK and [Demo source code](#) for your platform.

2. Click **Next**.



### Step 3. Configure demo project files

1. In the **Modify Configuration** step, select the development platform in line with the source package downloaded.
2. Find and open `TRTC-Simple-Demo/example/lib/debug/GenerateTestUserSig.dart`.
3. Set parameters in `GenerateTestUserSig.dart` as follows.
  - SDKAPPID: a placeholder by default. Set it to the actual `SDKAppID`.
  - `SECRETKEY`: a placeholder by default. Set it to the actual key.

- Click **Next** to complete the creation.
- After compilation, click **Return to Overview Page**.

- The method for generating `UserSig` described in this document involves configuring `SECRETKEY` in client code. In this method, `SECRETKEY` may be easily decompiled and reversed, and if your key is leaked, attackers can steal your Tencent Cloud traffic. Therefore, **this method is only suitable for the local execution and debugging of the demo.**
- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig`. For more information, please see [How do I calculate UserSig on the server?](#).

1. Run `flutter pub get` .
2. Compile, run, and debug the project.
  - Android
  - iOS



- Windows
- macOS
  - i. Run `flutter run` .
  - ii. Open the demo project with Android Studio (3.5 or above), and click **Run**.

## FAQs

### How do I view TRTC logs?

TRTC logs are compressed and encrypted by default with the `.xlog` extension at the following address:

- **iOS:** `Documents/log` in the sandbox.
- **Android:**
  - 6.7 or below: `/sdcard/log/tencent/liteav` .
  - 6.8 or above: `/sdcard/Android/data/package name/files/log/tencent/liteav/` .

### What should I do if videos do not show on iOS but do on Android?

Please check whether `io.flutter.embedded_views_preview` is `YES` in your `info.plist` .

### What should I do if Android Studio fails to build my project with the error "Manifest merge failed"?

Open `/example/android/app/src/main/AndroidManifest.xml` .

1. Add `xmlns:tools="http://schemas.android.com/tools"` to `manifest` .
2. Add `tools:replace="android:label"` to `application` .

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

Note :

For more FAQs, please see [Flutter](#).

# Unity

Last updated : 2022-03-30 18:19:54

This document shows how to integrate the TRTC SDK in Unity to enable audio/video calls in games.

The demo includes the following features:

- Room entry/exit
- Custom video rendering
- Device management and music/voice effects

Note :

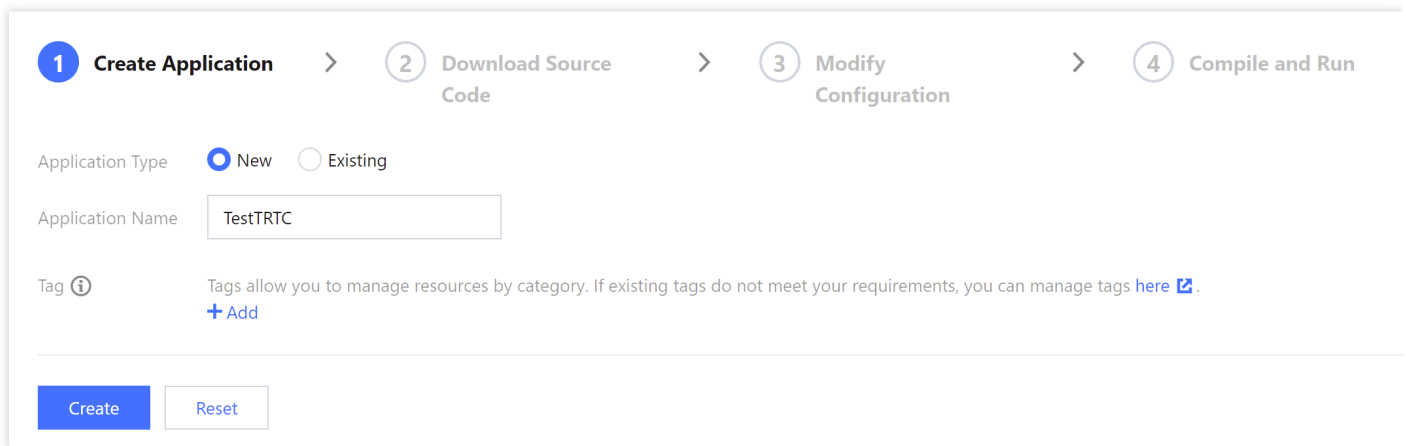
- For details about API features and parameters, please see [Client APIs > Unity > Overview](#).
- Unity 2020.2.1f1c1 is recommended.
- Supported platforms: Android, iOS, Windows, macOS (alpha testing)
- Modules required: `Android Build Support` , `iOS Build Support` , `Windows Build Support` , `MacOS Build Support`
- If you are developing for iOS, you also need:
  - Xcode 11.0 or above
  - A valid developer signature for your project

## Directions

### Step 1. Create an application

1. Log in to the TRTC console and select **Development Assistance** > [Demo Quick Run](#).

2. Enter an application name such as `TestTRTC` and click **Create**.



1 Create Application > 2 Download Source Code > 3 Modify Configuration > 4 Compile and Run

Application Type ☒ New ☐ Existing

Application Name

Tag ⓘ Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#). [+ Add](#)

Create Reset

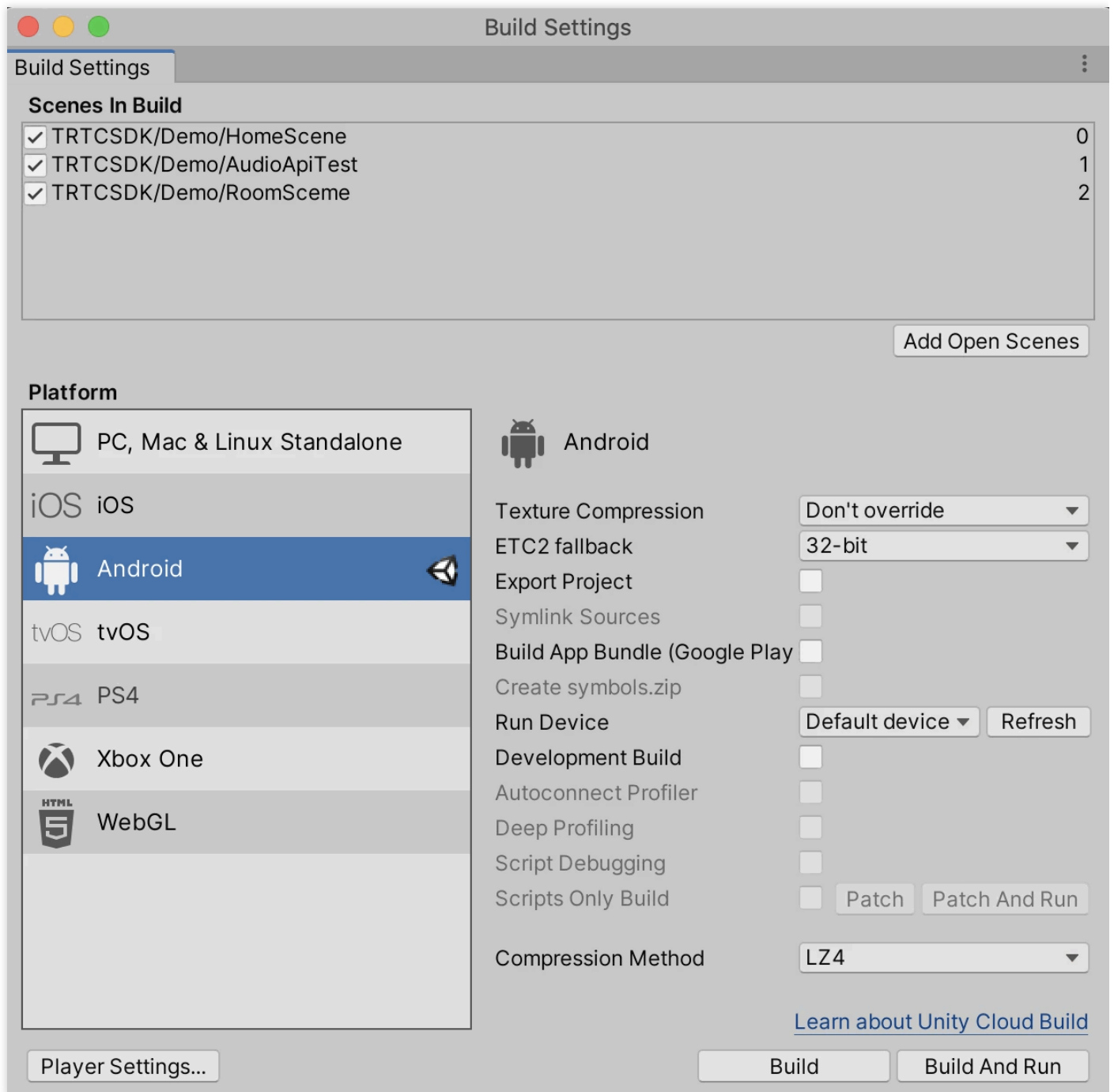
## Step 2. Download the SDK and source code

1. Download the SDK and [demo source code](#).
2. Click **Next**. You can open the project with Unity, or copy `TRTCUnitySDK/Assets/TRTCSDK/SDK` in the SDK ZIP file to the `Assets` directory of your project.
3. Find and open `Assets/TRTCSDK/Demo/Tools/GenerateTestUserSig.cs`.
4. Set parameters in `GenerateTestUserSig.cs` as follows:
  - SDKAPPID: ``0`` by default. Set it to the actual ``SDKAppID``.
  - SECRETKEY: left empty by default. Set it to the actual key.

## Step 3. Compile and run

- Android
- iOS
- Windows
- macOS

1. Open Unity Editor, go to **File > Build Settings**, and select **Android** for **Platform**.



2. Connect to a real Android device and click **Build And Run** to run the demo.
3. Call `enterRoom` first and go on to test other APIs. The data display window shows whether the call is successful, and the other window displays the callback information.

## Demo

The demo integrates most of the APIs launched so far, which can be used for testing and as reference for API calls. For more information about APIs, see [Client APIs > Unity > Overview](#).

Note :

The UI of the latest version of the demo may look different.

## Directory Structure

```
|—Assets
|   |— Editor // Unity Editor script
|   |   |— BuildScript.cs // Unity Editor build menu
|   |   |— IosPostProcess.cs // Script for building iOS application in Unity Editor
|   |— Plugins
|   |   |— Android
|   |   |   |— AndroidManifest.xml //Android configuration file
|   |— StreamingAssets // Audio/video stream files for the Unity demo
|   |— TRTCSDK
|   |— Demo // Unity demo
|   |— SDK // TRTC SDK for Unity
|   |   |— Implement // Implementation of TRTC SDK for Unity
|   |   |— Include // Header files of TRTC SDK for Unity
|   |   |— Plugins // Underlying implementation of TRTC SDK for Unity for different platforms
```

# React Native

Last updated : 2022-04-02 11:50:15

This document describes how to quickly run the TRTC demo for React Native.

## Environment Requirements

- React Native 0.63 or above
- Node (above v12) & Watchman
- **Developing for Android:**
  - Android Studio 3.5 or above
  - Devices with Android 4.1 or above
- **Developing for iOS and macOS:**
  - Xcode 11.0 or above
  - OS X 10.11 or above
  - A valid developer signature for your project
- For how to set up the environment, see the React Native [official document](#).

## Prerequisites

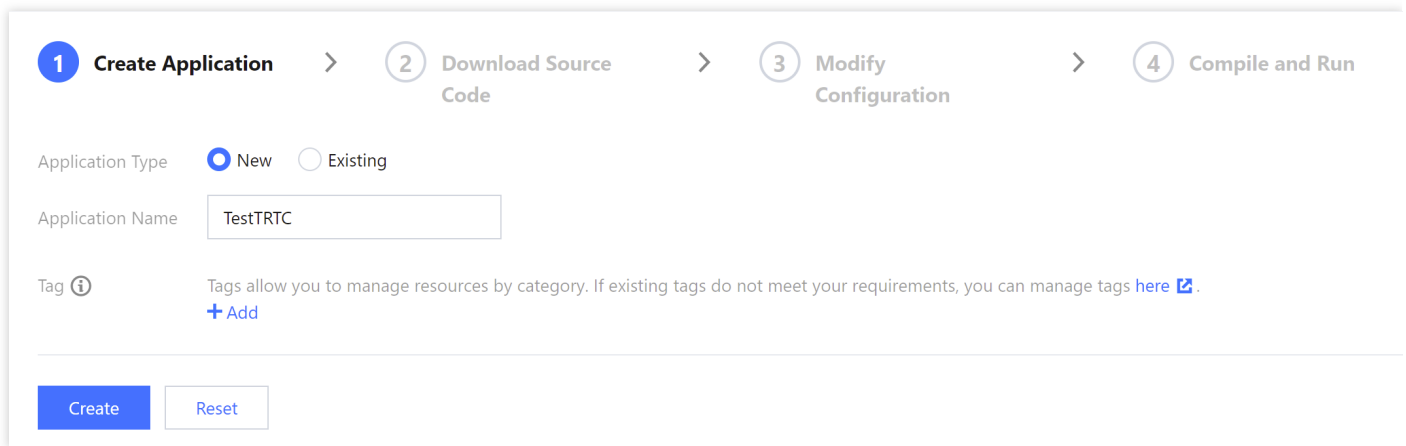
You have [signed up for a Tencent Cloud account](#) and verified your identity.

## Directions

### Step 1. Create an application

1. Log in to the TRTC console and select **Development Assistance** > **Demo Quick Run**.
2. Select **New** and enter an application name such as `TestTRTC` . If you have already created an application, select **Existing**.


3. Add or edit tags according to your actual business needs and click **Create**.



1 Create Application > 2 Download Source Code > 3 Modify Configuration > 4 Compile and Run

Application Type ☒ New ☐ Existing

Application Name

Tag ⓘ Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#) .

[+ Add](#)

Create Reset

Note :

- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to identify and organize your Tencent Cloud resources. For example, an enterprise may have multiple business units, each of which has one or more TRTC applications. In this case, the enterprise can tag TRTC applications to mark out the unit information. Tags are optional and can be added or edited according to your actual business needs.

## Step 2. Download the SDK and demo source code

1. Download the SDK and [demo source code](#) for your platform.
2. Click **Next**.

Note :

You cannot download the demo for React Native via the console at the moment. **Please download the source code via the above link.**

## Step 3. Configure demo project files

1. In the **Modify Configuration** step, select the development platform in line with the source package downloaded.



- Find and open `/debug/config.js`.
- Set the `SDKAPPID` and `SECRETKEY` parameters:
  - `SDKAPPID`: a placeholder by default. Set it to the actual ``SDKAppID``.
  - ``SECRETKEY``: a placeholder by default. Set it to the actual key.
- Click **Next** to complete the creation.
- After compilation, click **Return to Overview Page**.

Note :

- The method for generating `UserSig` described in this document involves configuring `SECRETKEY` in client code. In this method, `SECRETKEY` may be easily decompiled and reversed, and if your key is leaked, attackers can steal your Tencent Cloud traffic. Therefore, **this method is only suitable for the local execution and debugging of the demo.**
- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig`. For more information, see [How do I calculate UserSig on the server?](#).

## Step 4. Configure permission requests

You need to configure permission requests in order to run the demo.

- Android
- iOS

- Configure application permissions in `AndroidManifest.xml`. The TRTC SDK requires the following permissions:

```
<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" />
```

#### Note

Do not set `android:hardwareAccelerated="false"`. Disabling hardware acceleration will result in failure to render remote users' videos.

## 2. Manually configure audio and video permission requests.

```
if (Platform.OS === 'android') {
  await PermissionsAndroid.requestMultiple([
    PermissionsAndroid.PERMISSIONS.RECORD_AUDIO, //For audio calls
    PermissionsAndroid.PERMISSIONS.CAMERA, // For video calls
  ]);
}
```

## Step 5. Build and run the demo

Launch Metro and run `npx react-native start` under your React Native project directory.

- Android
- iOS

Open a new window and start debugging:

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

# Unreal Engine

Last updated : 2022-03-30 18:09:19

This document describes how to quickly run the TRTC demo for Unreal Engine.

Note :

Currently, the demo can be run on Windows, macOS, iOS, and Android.

## Environment Requirements

- Unreal Engine 4.27.1 or above
- **Developing for Android:**
  - Android Studio 4.0 or above
  - Visual Studio 2017 15.6 or above
  - A real device for testing
- **Developing for iOS and macOS:**
  - Xcode 11.0 or above
  - OS X 10.11 or above
  - A valid developer signature for your project
- **Developing for Windows:**
  - OS: Windows 7 SP1 or above (64-bit based on x86-64)
  - Disk space: at least 1.64 GB of space after the IDE and relevant tools are installed
  - [Visual Studio 2019](#)

## Prerequisites

You have [signed up for a Tencent Cloud account](#) and verified your identity.

## Directions

### Step 1. Create an application

1. In the TRTC console, select **Development Assistance** > [Demo Quick Run](#).

2. Enter an application name such as `TestTRTC` . If you have already created an application, click **Existing** to select it.
3. Add or edit tags according to your actual business needs and click **Create**.

1 Create Application > 2 Download Source Code > 3 Modify Configuration > 4 Compile and Run

Application Type ☒ New ☐ Existing

Application Name

Tag ⓘ Tags allow you to manage resources by category. If existing tags do not meet your requirements, you can manage tags [here](#).  
[+ Add](#)

Create Reset

Note :

- An application name can contain up to 15 characters. Only digits, letters, Chinese characters, and underscores are allowed.
- Tags are used to identify and organize your Tencent Cloud resources. For example, an enterprise may have multiple business units, each of which has one or more TRTC applications. In this case, the enterprise can tag TRTC applications to mark out the unit information. Tags are optional and can be added or edited according to your actual business needs.

## Step 2. Download the SDK and demo source code

1. Download the SDK and [demo source code](#). If you have any questions, create an issue [here](#).
2. Click **Next**.

## Step 3. Configure the demo project file

1. In the **Modify Configuration** step, select your platform.
2. Find and open `/TRTC_Demo/Source/debug/include/DebugDefs.h` .
3. Set parameters in `DebugDefs.h` as follows:

- SDKAPPID: `0` by default. Set it to the actual `SDKAppID`.
- SECRETKEY: left empty by default. Set it to the actual key.

Decompress the source package downloaded in Step 2, and open Android/TRTCSamplesDemo/debug/src/main/java/com/tencent/liteav/debug/GenerateTestUserSig.java File

**Paste SDKAppID and Secret Key to Specified Location**

SDKAppID

Secret Key

\* The secret key is sensitive information. Please do not disclose it.

```
public class GenerateTestUserSig {  
  
    private static final int SDKAPPID = 0;  
  
    private static final int EXPIRETIME = 604800;  
  
    private static final String SECRETKEY = "";
```

4. Click **Next** to complete the creation.
5. After compilation, click **Return to Overview Page**.

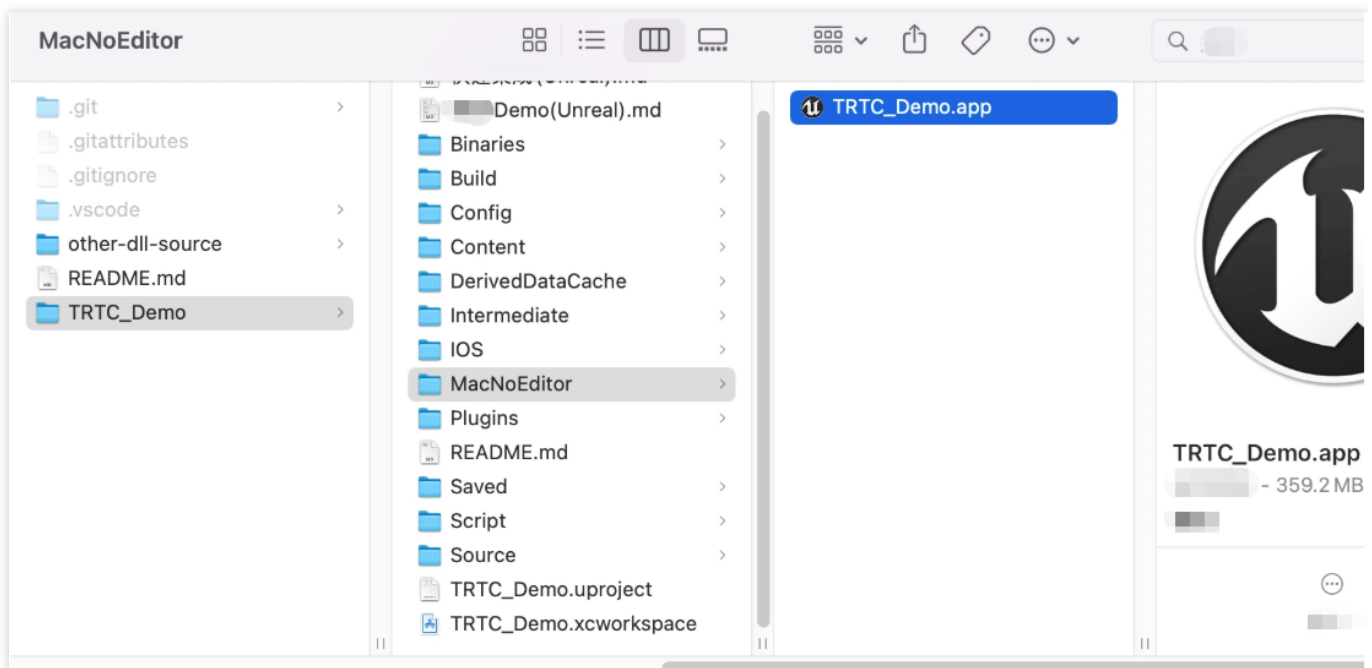
#### Note :

- The method for generating `UserSig` described in this document involves configuring `SECRETKEY` in client code. In this method, `SECRETKEY` may be easily decompiled and reversed, and if your key is leaked, attackers can steal your Tencent Cloud traffic. Therefore, **this method is only suitable for the local execution and debugging of the demo.**
- The correct `UserSig` distribution method is to integrate the calculation code of `UserSig` into your server and provide an application-oriented API. When `UserSig` is needed, your application can send a request to the business server for a dynamic `UserSig`. For more information, please see [How do I calculate UserSig on the server?](#).

## Step 4. Compile, package, and run the project

1. Open `/TRTC_Demo/TRTC_Demo.uproject`.
2. Compile, run, and test the project.

- macOS
- Windows
- iOS
- Android
- i. Go to **File > Package Project > Mac**.
- ii. Configure permissions. Right-click the `xxx.app` file compiled in the previous step and select **Show Package Contents**.



- iii. Go to **Contents > Info.plist**.
- iv. Select **Information Property List** and add the following two permissions:

```
<key>NSCameraUsageDescription</key>
<string>Video calls are possible only with camera permission.</string>
<key>NSMicrophoneUsageDescription</key>
<string>Audio calls are possible only with mic access.</string>
```

- v. If you use UE4 Editor, add the above permissions to the **UE4Editor.app** file.

## Running the Demo

The demo offers implementation code for one-to-one video calls, which you can use for testing or as reference for API calls. For more information about APIs, see [All Platforms \(C++\) > Overview](#).

Note :

The UI of the latest version of the demo may look different.



## TRTC Cross-Platform (C++) APIs

[API Documentation \(Chinese\)](#)

[API Documentation \(English\)](#)

## FAQs

### How do I view TRTC logs?

TRTC logs are compressed and encrypted by default with the `.xlog` extension. You can view them via the following paths:

- **iOS:** `Documents/Log` of the application sandbox

- **Android:**

- 6.7 or below: `/sdcard/log/tencent/liteav`
- 6.8 or above: `/sdcard/Android/data/package name/files/log/tencent/liteav/`

## What should I do if UE4 Editor crashes on macOS?

Make sure you have added the following audio/video permissions to `info.plist` of **UE4Editor.app**.

```
<key>NSCameraUsageDescription</key>
<string>Video calls are possible only with camera permission.</string>
<key>NSMicrophoneUsageDescription</key>
<string>Audio calls are possible only with mic access.</string>
```

## What should I do if the error "Attempt to construct staged filesystem reference from absolute path" occurs on Android?

Close the UE4 project, open Command Prompt, and type the following strings:

```
adb shell
cd sdcard
ls (you should see the UE4Game directory listed)
rm -r UE4Game
Compile your project again.
```



# FAQs for Beginners

Last updated : 2022-03-18 22:40:44

## What is UserSig?

`UserSig` is a security signature designed by Tencent Cloud to prevent attackers from accessing your Tencent Cloud account.

Currently, Tencent Cloud services including TRTC, IM, and MLVB all use this security mechanism. Whenever you want to use these services, you must provide three key pieces of information, i.e.

`SDKAppID` , `UserID` , and `UserSig` in the initialization or login function of the corresponding SDK.

`SDKAppID` is used to identify your application, and `UserID` your user. `UserSig` is a security signature calculated based on the two parameters using the **HMAC SHA256** encryption algorithm. Attackers cannot use your Tencent Cloud traffic without authorization as long as they cannot forge a `UserSig` .

See the figure below for how `UserSig` is calculated. Basically, it involves hashing crucial information such as `SDKAppID` , `UserID` , and `ExpireTime` .

```
// UserSig formula, in which `secretkey` is the key used to calculate UserSig
usersig = hmacsha256(secretkey, (userid + sdkappid + currtime + expire +
base64(userid + sdkappid + currtime + expire)))
```

Note :

- `currtime` is the current system time and `expire` the expiration time of the signature.
- For detailed directions on how to calculate and get `UserSig` , please see [UserSig](#).

## How many rooms can there be in TRTC at the same time?

There can be up to 4,294,967,294 concurrent rooms in TRTC. No limits are set on the number of non-concurrent rooms.

## How long is the average delay in TRTC?

The average end-to-end delay of TRTC around the globe is less than 300 ms.

## Does TRTC support screen sharing on PCs?

Yes. For details, see the following documents:

- [Real-Time Screen Sharing \(Windows\)](#)

- [Real-Time Screen Sharing \(macOS\)](#)
- [Real-Time Screen Sharing \(Web\)](#)

For more information on the screen sharing APIs, please see [Client APIs > All Platforms \(C++\) > Overview](#) or [Client APIs > Electron > Overview](#).

## What platforms does TRTC support?

TRTC supports platforms including iOS, Android, Windows (C++), Windows (C#), macOS, web, and Electron. For more information, see [Supported Platforms](#).

## How many people can there be in a TRTC call?

- In call scenarios, each room can accommodate up to 300 concurrent users, and up to 50 of them can turn on their cameras or mics.
- In live streaming scenarios, each room can accommodate up to 100,000 concurrent users, and up to 50 of them can be assigned the anchor role and turn on their cameras or mics.

## How do I start a live streaming session in TRTC?

TRTC offers a dedicated low-latency interactive live streaming solution that allows up to 100,000 participants with co-anchoring latency kept as low as 200 ms and watch latency below 1s. It adapts excellently to poor network conditions and is optimized for the complicated mobile network environments.

For detailed directions, please see [Live Streaming Mode](#).

## What roles are supported during live streaming in TRTC? How do they differ from each other?

The live streaming scenarios ( `TRTCAppSceneLIVE` and `TRTCAppSceneVoiceChatRoom` ) support two roles: `TRTCRoleAnchor` (anchor) and `TRTCRoleAudience` (audience). An anchor can both send and receive audio/video data, but audience can only receive and play back others' data. You can call `switchRole()` to switch roles.

## Can I kick a user out, forbid a user to speak, or mute a user in a TRTC room?

Yes, you can.

- To enable the features through simple signaling operations, use `sendCustomCmdMsg`, the custom signaling API of TRTC, to define your own control signaling, and users who receive the message will perform the action expected. For example, to kick out a user, just define a kick-out signaling, and the user receiving it will exit the room.
- If you want to implement a more comprehensive operation logic, we recommend that you use [Instant Messaging](#) to map the TRTC room to an IM group and enable the features via the

sending/receiving of custom messages in the group.

## Can TRTC pull and play back streams through CDN?

Yes. For details, please see [CDN Relayed Live Streaming](#).

## Does TRTC support Swift integration on iOS?

Yes. Just integrate the SDK in the same steps as you do a third-party library or by following the steps in [Demo Quick Start \(iOS & macOS\)](#).

## What browsers does the SDK for web support?

It is well supported by Chrome (desktop) and Safari (desktop and mobile) but poorly or not supported by other platforms such as browsers on Android. For more information, please see [Client APIs > Supported Platforms](#).

You can open [WebRTC Support Level Test](#) in a browser to test whether the environment fully supports WebRTC.

## What do the errors `NotFoundError` , `NotAllowedError` , `NotReadableError` , `OverConstrainedError` , and `AbortError` found in the log of TRTC SDK for web mean?

Error	Description	Suggested Solution
<code>NotFoundError</code>	The media (audio, video, or screen sharing) of the request parameters are not found. For example, this error occurs if the PC has no cameras but the browser requests a video stream.	Remind users to check devices such as cameras and mics before making a call. If a user does not have a camera and wants to make an audio call, use <code>TRTC.createStream({ audio: true, video: false })</code> to make the SDK capture audio only.
<code>NotAllowedError</code>	The user has rejected the request of the current browser instance to access the camera/mic or share screens.	Remind the user that audio/video calls are not possible without camera/mic access.

Error	Description	Suggested Solution
NotReadableError	The user has granted access to the requested device, but it is still inaccessible due to a hardware, browser or webpage error.	Handle the error according to the error message returned, and send this message to the user: "The camera/mic cannot be accessed. Please make sure that no other applications are requesting access and try again."
OverConstrainedError	The <code>cameraId/microphoneId</code> value is invalid.	Make sure that the <code>cameraId/microphoneId</code> value passed in is valid.
AbortError	The device cannot be accessed due to an unknown reason.	-

For more information, please see [initialize](#).

## How do I check whether TRTC SDK for web can get the device (camera/mic) list?

### 1. Check whether the browser can access the devices:

Open the console with the browser and enter `navigator.mediaDevices.enumerateDevices()` to see if the device list can be obtained.

- Normally, a promise containing an array of `MediaDeviceInfo` objects will be returned, each object corresponding to an available media device.
- If the SDK fails to enumerate the devices, a rejected promise will be returned, indicating that the browser fails to detect any devices. You need to check the browser or devices.

### 2. If the device list can be obtained, enter `navigator.mediaDevices.getUserMedia({ audio: true, video: true })` to see if the `MediaStream` object can be returned. If it is not returned, it indicates that the browser failed to obtain any data. You need to check your browser configuration.

## How do live streaming, interactive live streaming, TRTC, and relayed live streaming differ from and relate to each other?

- **Live streaming** (keywords: one-to-many, RTMP/HLS/HTTP-FLV, CDN)

Live streaming consists of the push end, the playback end, and the cloud live streaming service.

Streams are pushed over the universal protocol RTMP, delivered through CDNs, and can be watched over protocols including RTMP, HTTP-FLV, or HLS (for HTML5).

- **\*Interactive live streaming\*** (keywords: co-anchoring, anchor competition)

In interactive live streaming, audience can co-anchor with anchors and anchors from different rooms can compete with each other.

- **Real-time communication** (keywords: multi-person interaction, UDP-based proprietary protocol, low latency)

The main application scenarios for TRTC (Tencent Real-Time Communication) are audio/video interaction and low-latency live streaming. It uses a UDP-based proprietary protocol and can keep the latency as low as 100 ms. Typical applications include QQ calls, VooV Meeting, and online group classes. TRTC is supported by mainstream platforms including iOS, Android, and Windows and can communicate over WebRTC. It supports relaying streams to CDNs through on-cloud stream mixing.

- **Relayed live streaming** (keywords: on-cloud stream mixing, RTC relayed live streaming, CDN)

The relayed live streaming technology replicates multiple streams in a low-latency co-anchoring room and mixes them into one stream in the cloud before pushing it to a live streaming CDN for delivery and playback.

## How do I view my call duration and usage?

You can find the information on the [Usage Statistics](#) page of the TRTC console.

## How do I fix stutter?

You can check call quality by room ID or user ID in [Monitoring Dashboard](#) in the TRTC console.

- Check the send and receive statistics from the recipient's perspective.
- Check the send and receive packet loss. High packet loss suggest that the stutter may be caused by unstable network connections.
- Check the frame rate and CPU usage. Both low frame rates and high CPU usage can cause stutter.

## How do I fix low-quality, blurry and pixelated videos?

- Resolution is mainly associated with bitrate. Check whether the bitrate is set too low. Pixelation tends to occur when resolution is high but bitrate low.
- TRTC dynamically adjusts bitrate and resolution based on network conditions according to its on-cloud QoS control policy. It reduces the bitrate in case of poor network connections, which leads to decreased definition.
- Check whether the `VideoCall` or `Live` mode is used during room entry. As the `VideoCall` mode is designed for calls and features low latency and smoothness, it tends to sacrifice video quality

for smoothness when network connections are poor. We recommend that you use the `Live` mode for application scenarios with high requirements on video quality.

## How do I view the latest version number of the SDK?

- In the case of automatic loading, `latest.release` will load the latest version automatically. You don't need to modify the version number. For detailed instructions on integration, please see [SDK Quick Integration](#).
- You can find the latest version number of the SDK on the release notes page.
  - For iOS & Android, please see [Release Notes \(App\)](#).
  - For web, please see [Release Notes \(Web\)](#).
  - For Electron, please see [Release Notes \(Electron\)](#).