Mobile Live Video Broadcasting

SDK Download

Product Documentation

Contents

SDK Download

Download

Last updated：2022-09-14 17:52:39

We offer the MLVB Professional SDK for users outside the Chinese mainland. To learn about its features and the license required to use it, see Features.

MLVB Professional

MLVB Professional integrates multiple core audio/video features of Tencent Cloud, including MLVB, TRTC, live publishing/playback, and basic filters.

Platform	ZIP File	64-bit Support	Increase Installation Package By	Downsizing Installation Package
iOS	Download	Yes	4.08 MB (arm64)	Document
Android	Download	Yes	jar: 1.5 MB so (armeabi): 6.5 MB so (armv7): 6.1 MB so (arm64): 7.3 MB	Document

Note：
Different authorization is required to use different features of MLVB Professional:

- To use MLVB features, you must purchase an MLVB Professional license.
- To use TRTC features, you must purchase a TRTC package.
- Before you use the MLVB SDK, you need to set up the GDPR environment:

Android：

```
V2TXLivePremier.setEnvironment("GDPR");
```

iOS：

```
[V2TXLivePremier setEnvironment:@"GDPR"];
```

SDK Integration
iOS

Last updated：2022-09-14 17:14:53

This document describes how to quickly integrate RT-Cube’s MLVB LiteAVSDK 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 SDK

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

CocoaPods

1. Install CocoaPods

Enter the following command in a terminal window (you need to install Ruby on your macOS 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 path of the PODSPEC file of LiteAVSDK

```
platform :ios, '9.0'

target 'App' do
```

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

- Method 2: use CocoaPod’s official source, which allows version selection

4. Update the local repository and install the SDK

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

```
pod install
```

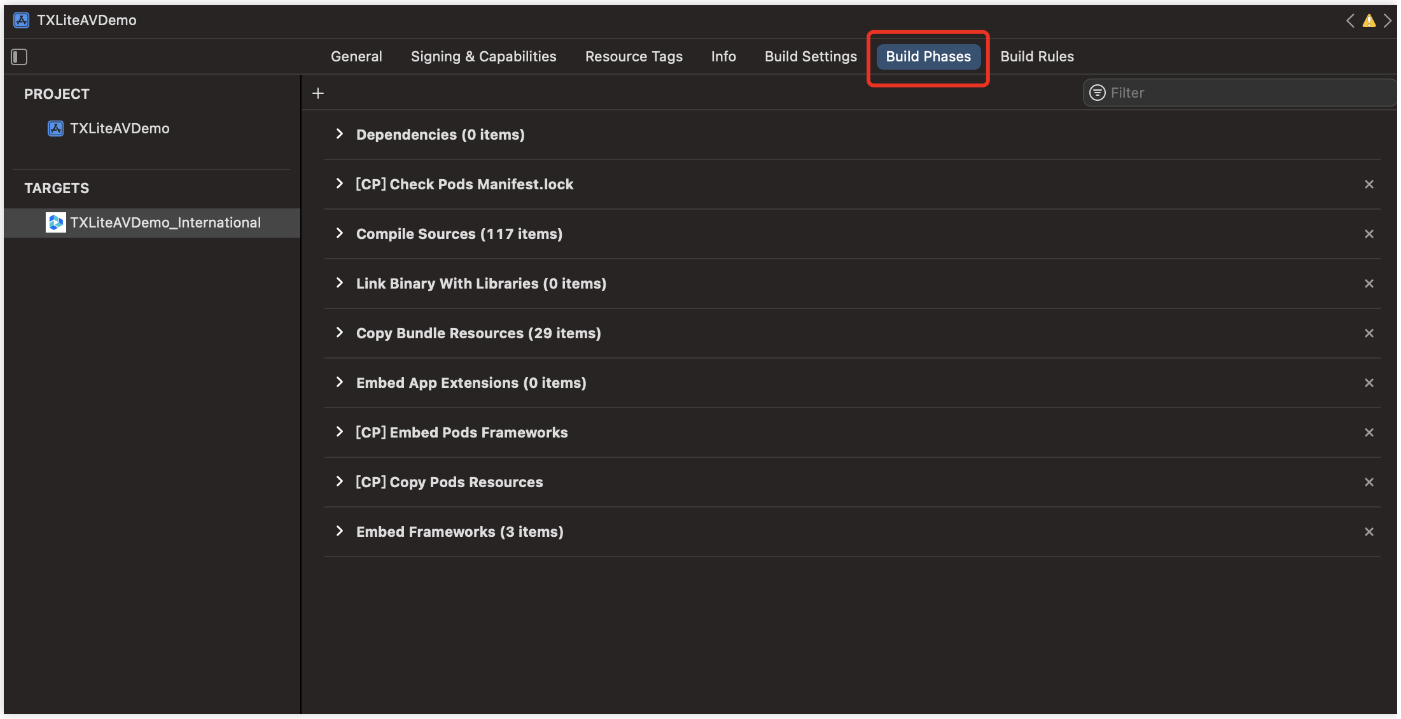
Or, run the following command to update the local repository:

```
pod update
```

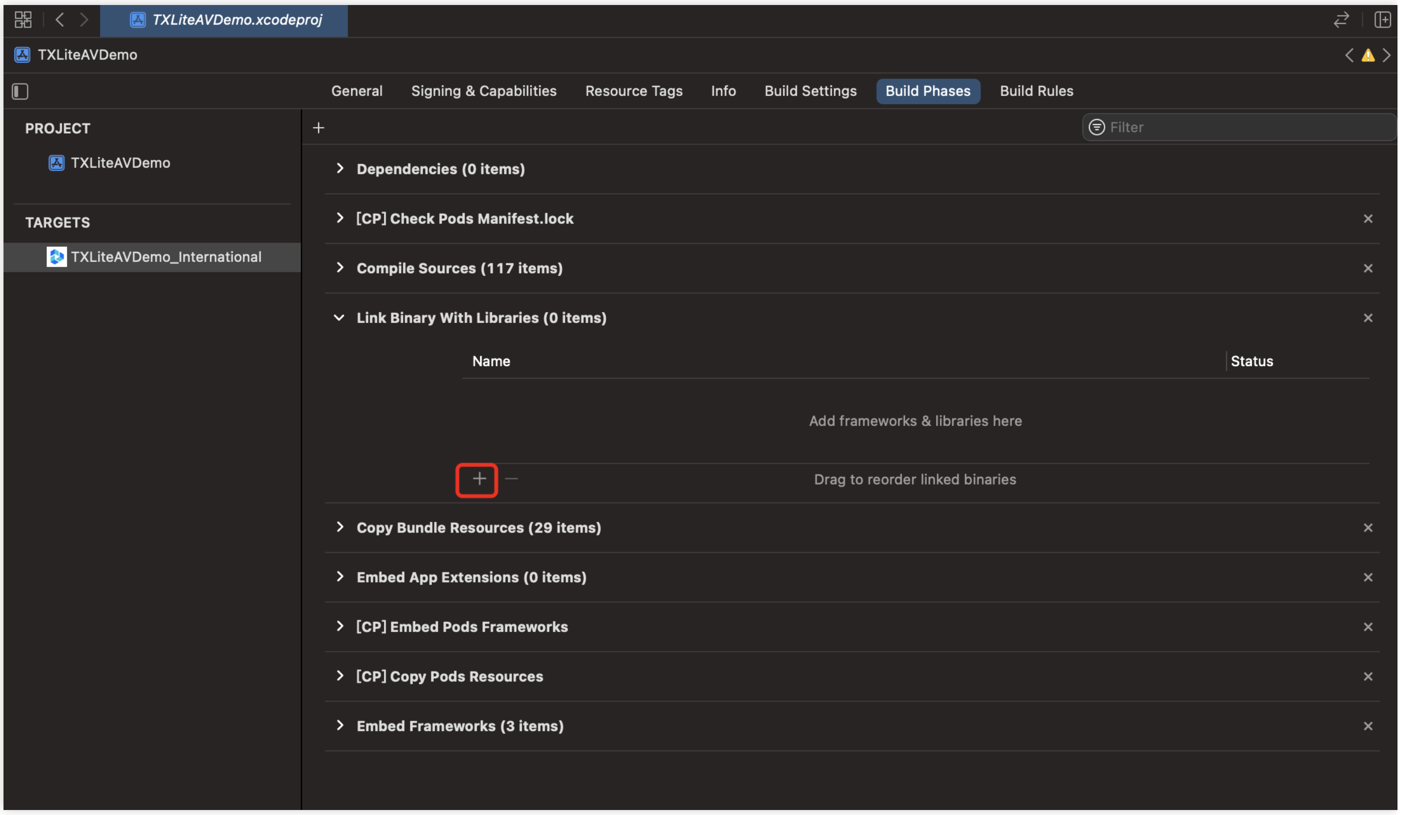
An XCWORKSPACE project file integrated with LiteAVSDK will be generated. Double-click to open the file.

Manual integration

1. Download LiteAVSDK and decompress the file.
2. Open your Xcode project, select the target you want to run, and select **Build Phases**.

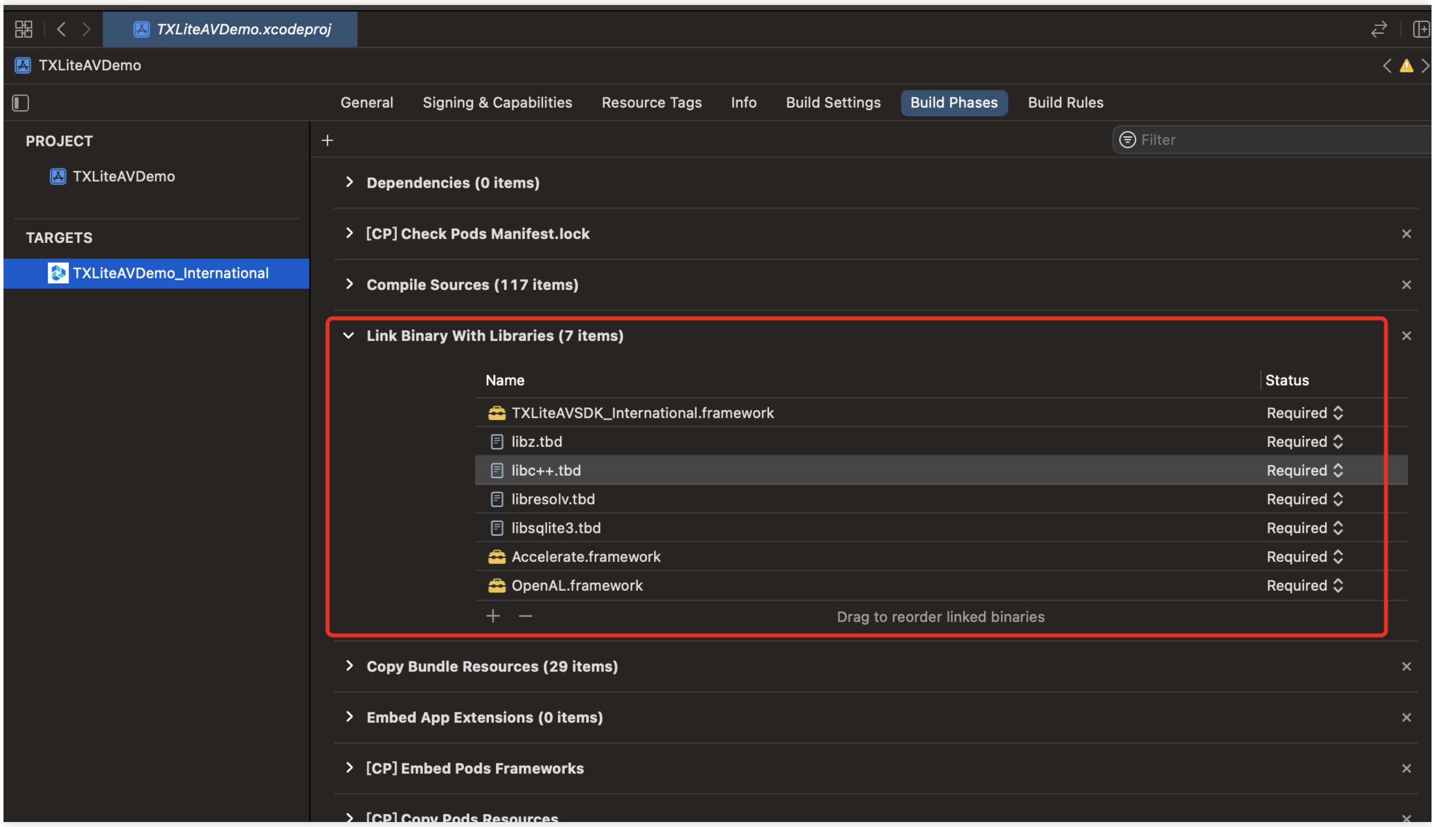

TXLiteAVDemo
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
TXLiteAVDemo
TARGETS
TXLiteAVDemo_International
Filter
Dependencies (0 items)
[CP] Check Pods Manifest.lock
Compile Sources (117 items)
Link Binary With Libraries (0 items)
Copy Bundle Resources (29 items)
Embed App Extensions (0 items)
[CP] Embed Pods Frameworks
[CP] Copy Pods Resources
Embed Frameworks (3 items)

3. Expand **Link Binary with Libraries** and click **+** at the bottom to add the libraries to depend on.


TXLiteAVDemo.xcodeproj
TXLiteAVDemo
General Signing & Capabilities Resource Tags Info Build Settings Build Phases Build Rules
PROJECT
TXLiteAVDemo
TARGETS
TXLiteAVDemo_International
Dependencies (0 items)
[CP] Check Pods Manifest.lock
Compile Sources (117 items)
Link Binary With Libraries (0 items)
Name Status
Add frameworks & libraries here
Drag to reorder linked binaries
Copy Bundle Resources (29 items)
Embed App Extensions (0 items)
[CP] Embed Pods Frameworks
[CP] Copy Pods Resources
Embed Frameworks (3 items)

4. Add the downloaded `TXLiteAVSDK_Professional.framework` and the libraries it depends on:

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

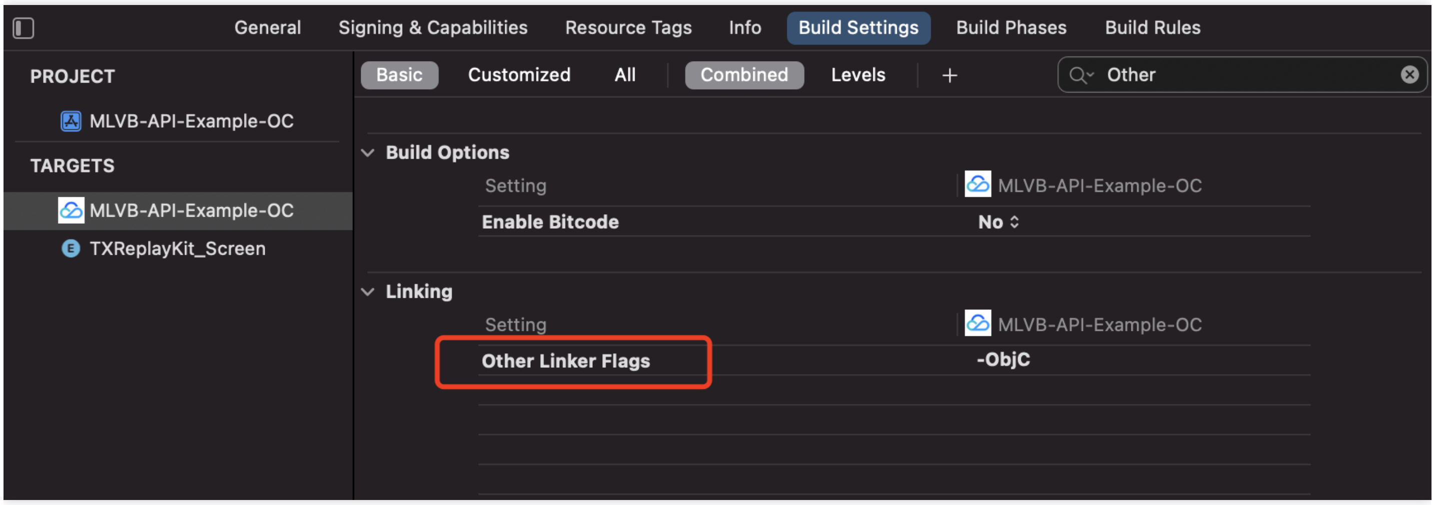


Link Binary With Libraries (7 items)

Name	Status
TXLiteAVSDK_International.framework	Required
libz.tbd	Required
libc++.tbd	Required
libresolv.tbd	Required
libsqlite3.tbd	Required
Accelerate.framework	Required
OpenAL.framework	Required

Drag to reorder linked binaries

5. Click **Build Settings**, search for `Other Linker Flags` , and add `-ObjC` .



Build Options

Setting	MLVB-API-Example-OC
Enable Bitcode	No

Linking

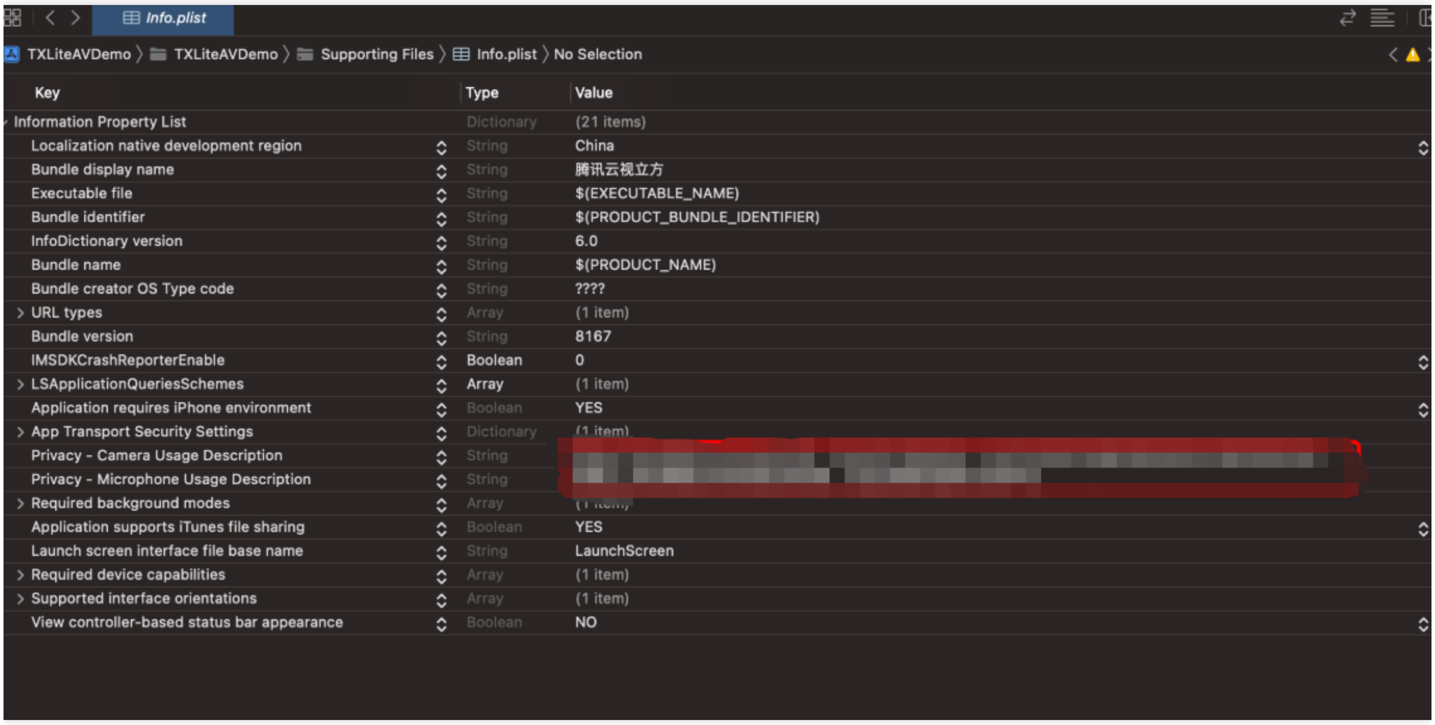
Setting	MLVB-API-Example-OC
Other Linker Flags	-ObjC

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 to display pop-up messages asking for mic and camera permissions.

- **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


Info.plist

TXLiteAVDemo 〉 TXLiteAVDemo 〉 Supporting Files 〉 Info.plist 〉 No Selection

Key	Type	Value
Information Property List	Dictionary	(21 items)
Localization native development region	String	China
Bundle display name	String	腾讯云视立方
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle creator OS Type code	String	????
URL types	Array	(1 item)
Bundle version	String	8167
IMSDKCrashReporterEnable	Boolean	0
LSApplicationQueriesSchemes	Array	(1 item)
Application requires iPhone environment	Boolean	YES
App Transport Security Settings	Dictionary	(1 item)
Privacy - Camera Usage Description	String	[redacted]
Privacy - Microphone Usage Description	String	[redacted]
Required background modes	Array	(1 item)
Application supports iTunes file sharing	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
Required device capabilities	Array	(1 item)
Supported interface orientations	Array	(1 item)
View controller-based status bar appearance	Boolean	NO

Importing the SDK

There are two ways to import the SDK in your project code.

- **Method 1:** import the SDK module in the files that need to use the SDK's APIs in your project

```
@import TXLiteAVSDK_Professional;
```

- **Method 2:** import a specific header file in the files that need to use the SDK's APIs in your project

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

Configuring License

Log in to the CSS console, go to **MLVB SDK** > **License Management**, and click **Get License** to obtain a trial license. For detailed directions, see Applying for trial license. You will get two strings: a license URL and a decryption key.

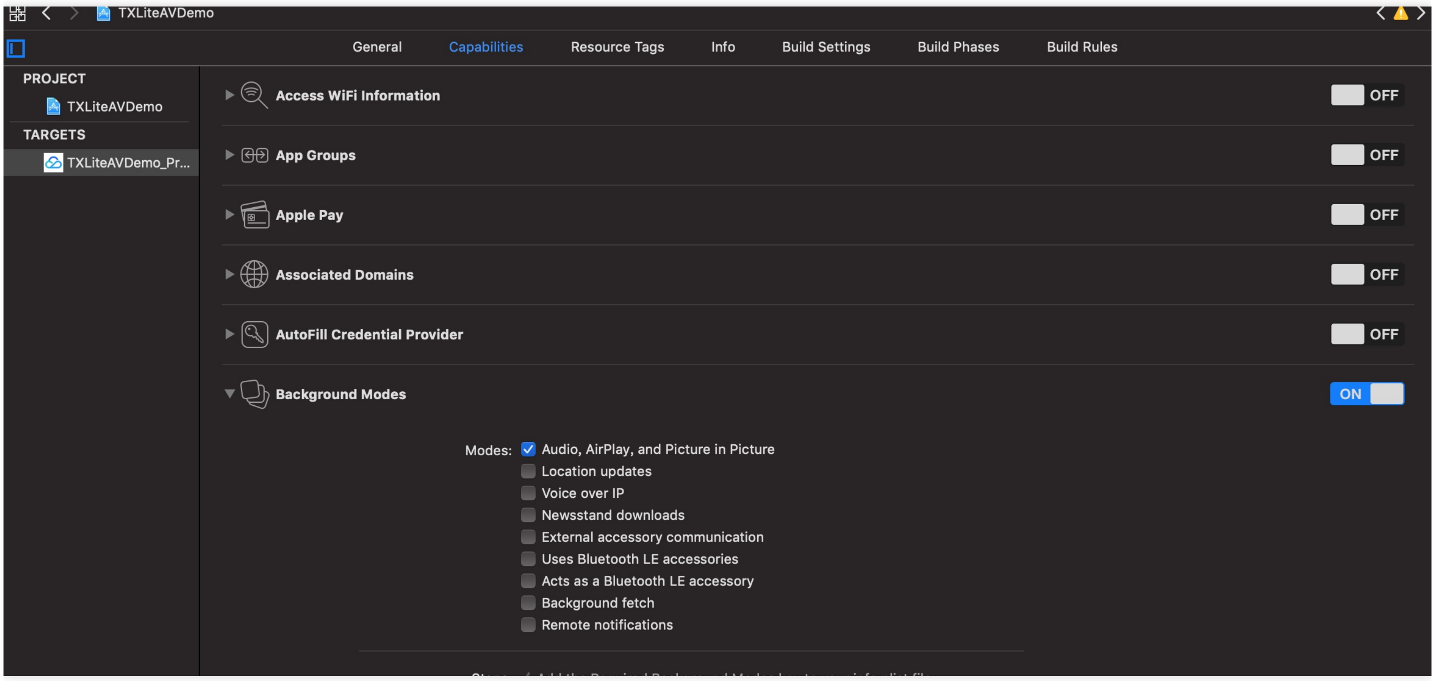
Before you use LiteAVSDK features in your application, complete the following configurations (preferably in `- [AppDelegate application:didFinishLaunchingWithOptions:]`):

```
@import TXLiteAVSDK_Professional;
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(N
SDictionary *)launchOptions {
NSString * const licenceURL = @"<The license URL obtained>";
NSString * const licenceKey = @"<The key obtained>";

//TXLiveBase can be found in the "TXLiveBase.h" header file
[TXLiveBase setLicenceURL:licenceURL key:licenceKey];
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
@end
```

FAQs

Can I run LiteAVSDK in the background?

Yes, you can. If you want the SDK to run in the background, select your project, under the **Capabilities** tab, set **Background Modes** to **ON**, and check **Audio, AirPlay and Picture in Picture**, as shown below:



Access WiFi Information: OFF
App Groups: OFF
Apple Pay: OFF
Associated Domains: OFF
AutoFill Credential Provider: OFF
Background Modes: ON

Modes:
- [✓] Audio, AirPlay, and Picture in Picture
- [ ] Location updates
- [ ] Voice over IP
- [ ] Newsstand downloads
- [ ] External accessory communication
- [ ] Uses Bluetooth LE accessories
- [ ] Acts as a Bluetooth LE accessory
- [ ] Background fetch
- [ ] Remote notifications

Android

Last updated：2022-09-14 17:17:02

This document describes how to quickly integrate Tencent Cloud LiteAVSDK for Android into your project.

Environment Requirements

- Android Studio 2.0 or above
- Android 4.1 (SDK API level 16) or above

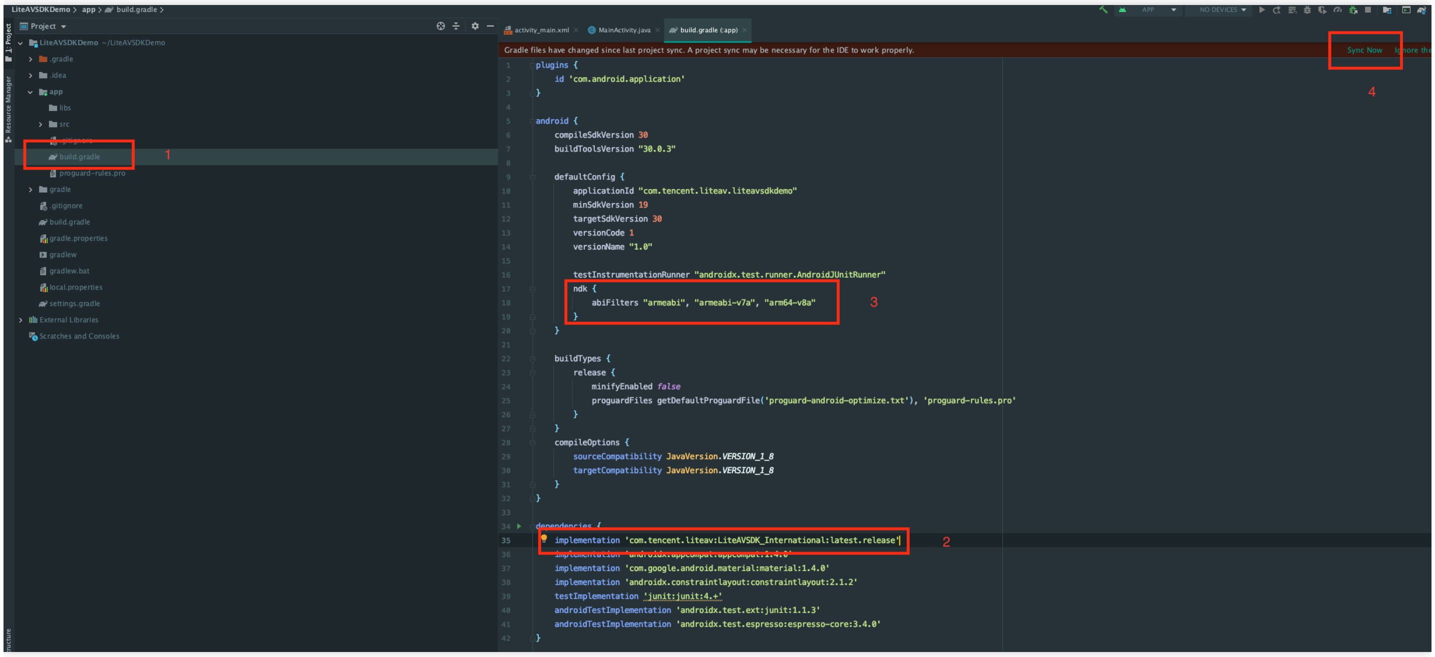
Integrating the SDK (AAR)

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)

Since JCenter has been deprecated, you can configure a Maven Central repository in Gradle to automatically download and update LiteAVSDK.

Open your project with Android Studio and modify the `build.gradle` file as described below to complete the integration.



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
plugins {
    id 'com.android.application'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.tencent.liteav.liteavsdkdemo"
        minSdkVersion 19
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        ndk {
            abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation 'com.tencent.liteav:LiteAVSDK_International:latest.release'
    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

1. Open `build.gradle` under your application.
2. Add the LiteAVSDK dependency to `dependencies` .

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

Or

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

3. In `defaultConfig` , specify the CPU architecture to be used by the application. Currently, LiteAVSDK supports armeabi, armeabi-v7a, and arm64-v8a.

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

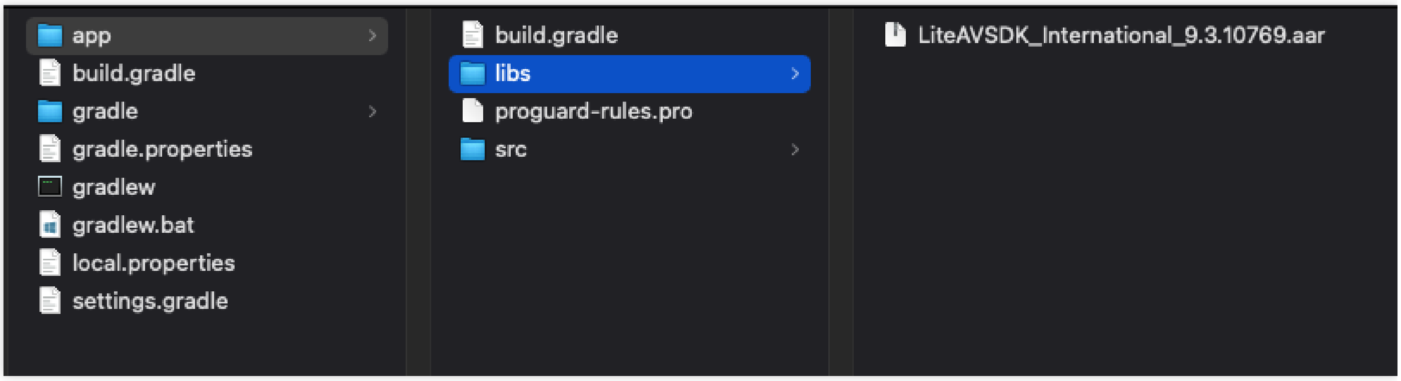
4. Click the **Sync Now** button to sync the SDK. If you have no problem accessing Maven Central, the SDK will be downloaded and integrated into your project automatically.

Method 2: manual download (AAR)

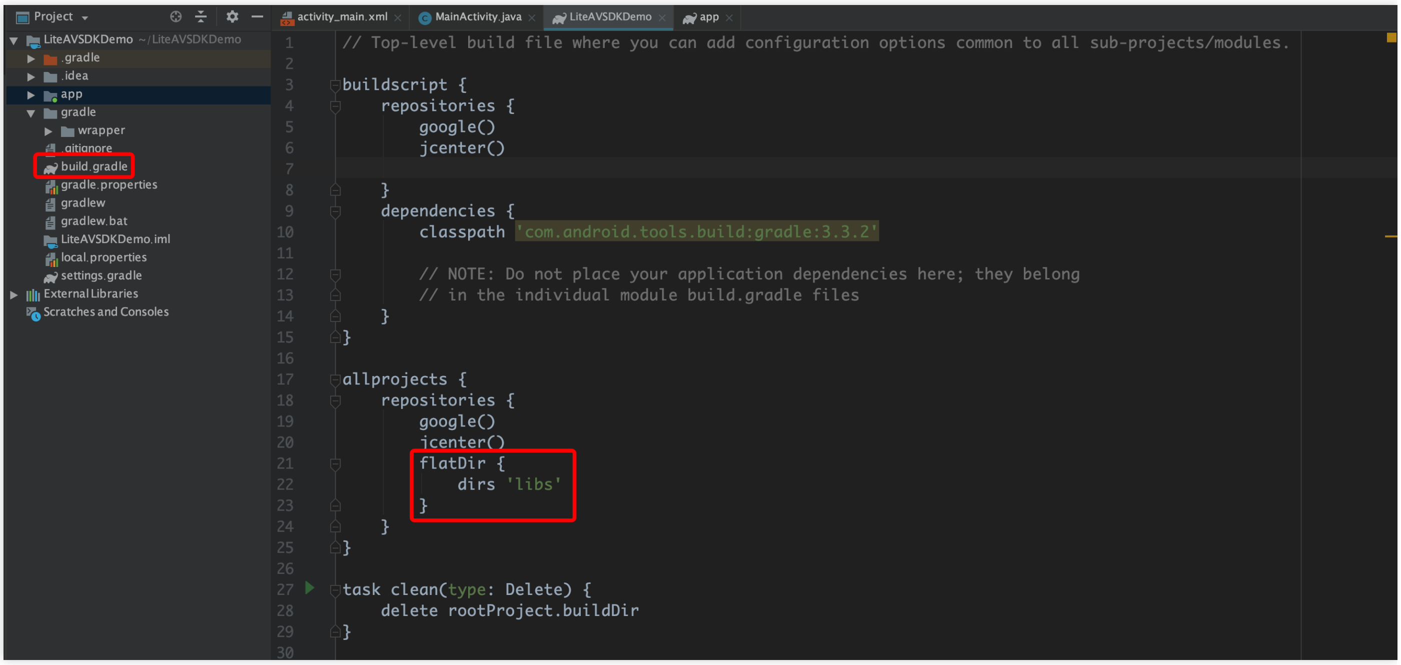
If you have problem accessing Maven Central, you can manually download the SDK and integrate it into your project.

1. Download LiteAVSDK and decompress the file.

2. Copy the AAR file in the SDK directory to the **app/libs** directory of your project.

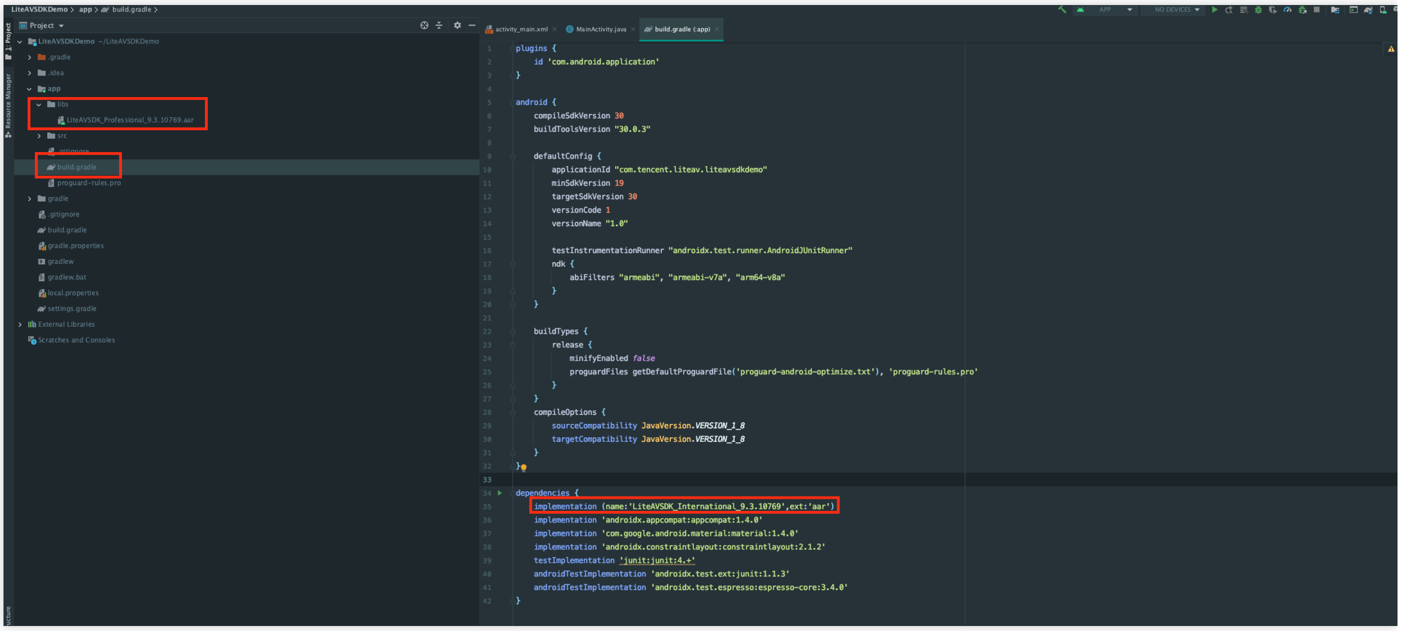


3. Add **flatDir** to `build.gradle` under your project's root directory to specify the local path for the repository.



```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.3.2'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        flatDir {
            dirs 'libs'
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

4. Add the LiteAVSDK dependency and, in `app/build.gradle`, add code that references the AAR file.



```
plugins {
    id 'com.android.application'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.tencent.liteav.liteavsdkdemo"
        minSdkVersion 19
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        ndk {
            abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation (name:'LiteAVSDK_International_9.3.10769',ext:'aar')
    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

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

5. In `defaultConfig` of `app/build.gradle`, specify the CPU architecture to be used by the application. Currently, LiteAVSDK supports armeabi, armeabi-v7a, and arm64-v8a.

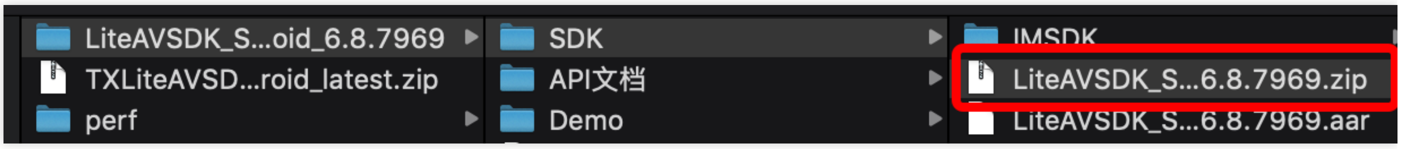
```
defaultConfig {
ndk {
abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
}
}
```

6. Click **Sync Now** to complete the integration of LiteAVSDK.

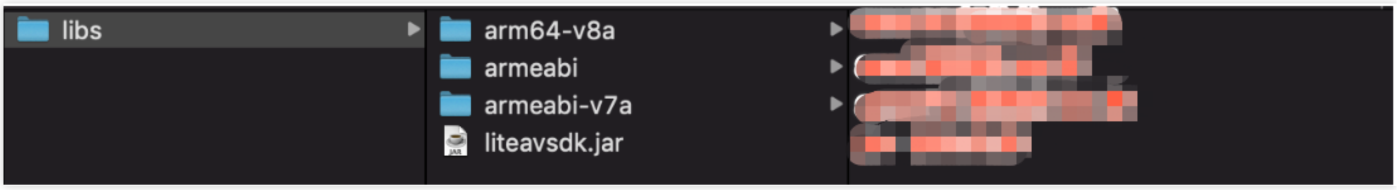
Integrating the SDK (JAR)

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

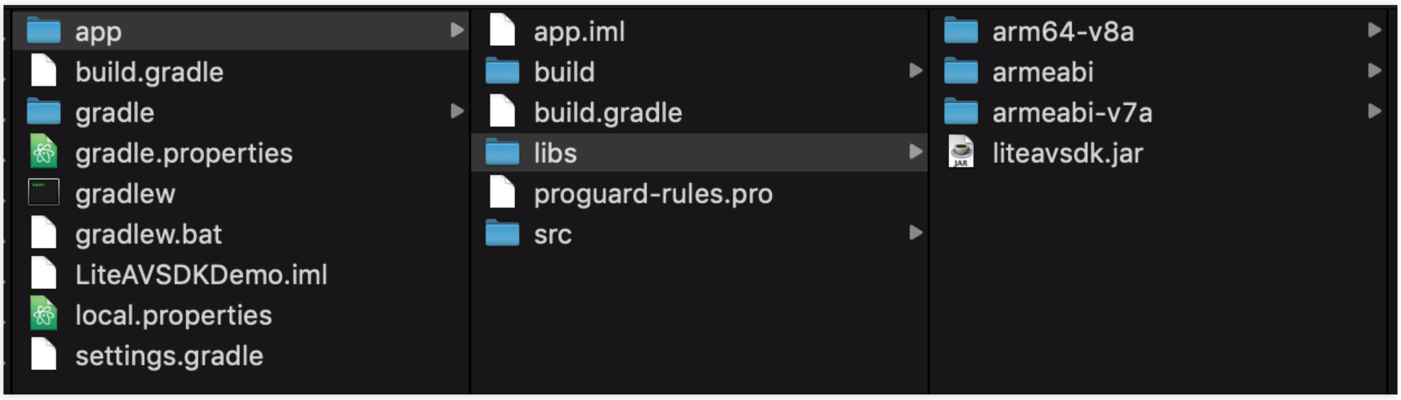
1. Download LiteAVSDK and decompress the file. In the SDK directory, find `LiteAVSDK_Professional_xxx.zip` (`xxx` indicates the version number of LiteAVSDK).


LiteAVSDK_S...oid_6.8.7969
TXLiteAVSD...roid_latest.zip
perf
SDK
API文档
Demo
IMSDK
LiteAVSDK_S...6.8.7969.zip
LiteAVSDK_S...6.8.7969.aar

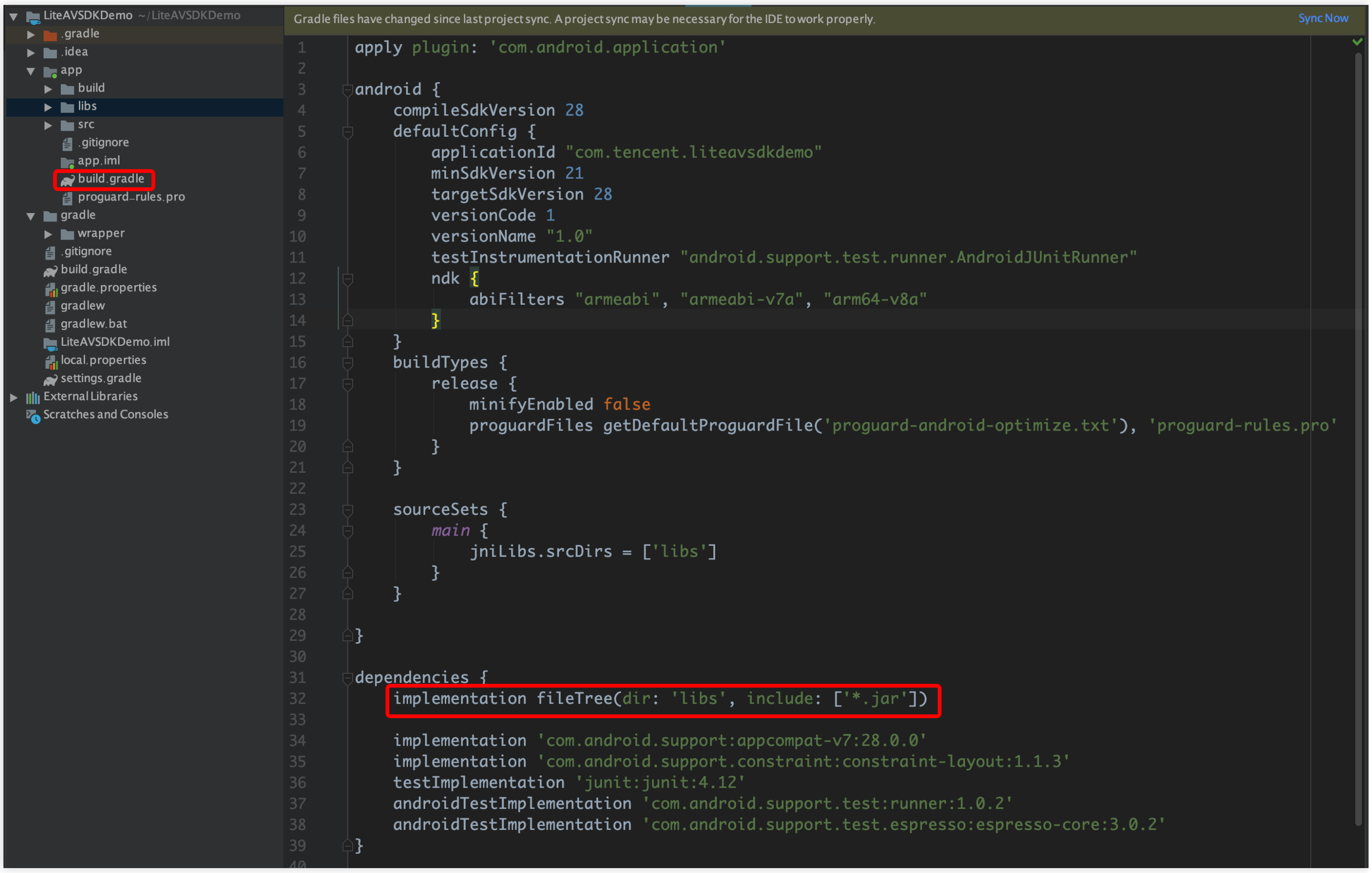
Decompress the file, and you will find a `libs` directory that contains a JAR file and several SO folders, as shown below:



2. Copy the JAR file and `armeabi`, `armeabi-v7a`, and `arm64-v8a` folders to the `app/libs` directory.



3. Add code that references the JAR library in `app/build.gradle` .

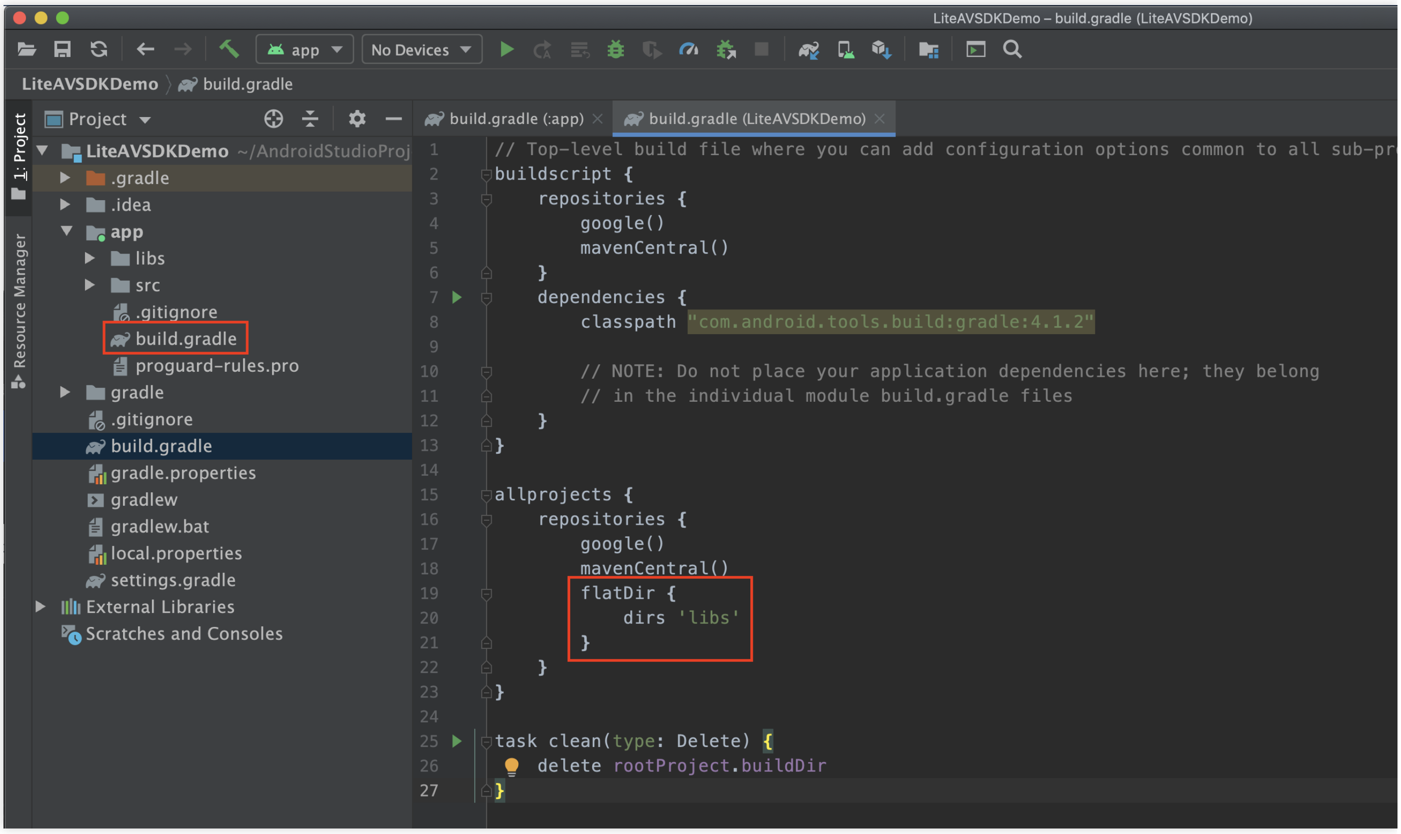


Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.tencent.liteavsdkdemo"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        ndk {
            abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }

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

}

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

    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

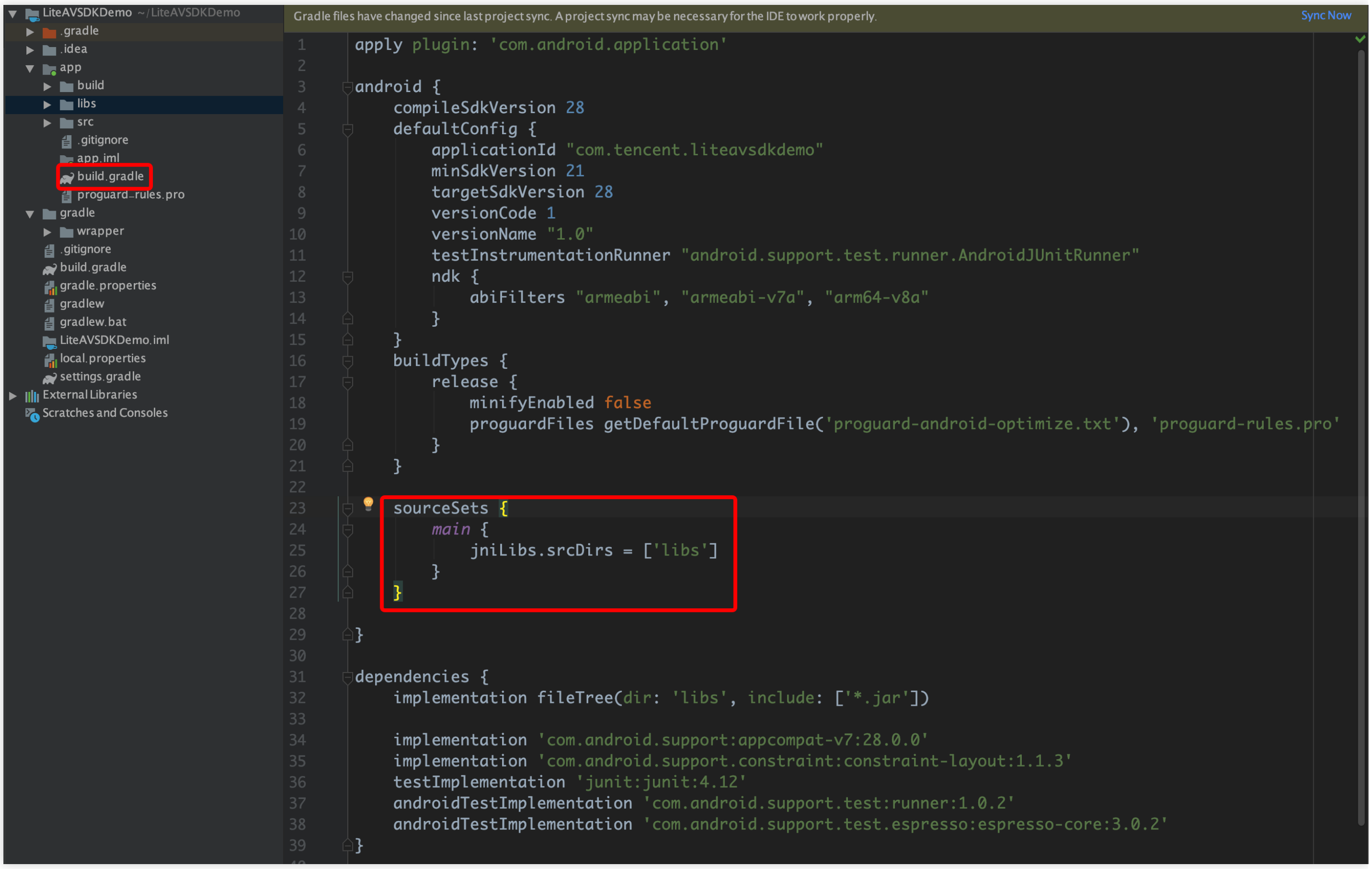
```
dependencies {
implementation fileTree(dir:'libs',include:['*.jar'])
}
```

4. Add **flatDir** to `build.gradle` under the project's root directory to specify the local path for the repository.



```
// Top-level build file where you can add configuration options common to all sub-pr
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:4.1.2"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        mavenCentral()
        flatDir {
            dirs 'libs'
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

5. In `app/build.gradle` , add code that references the SO libraries.



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

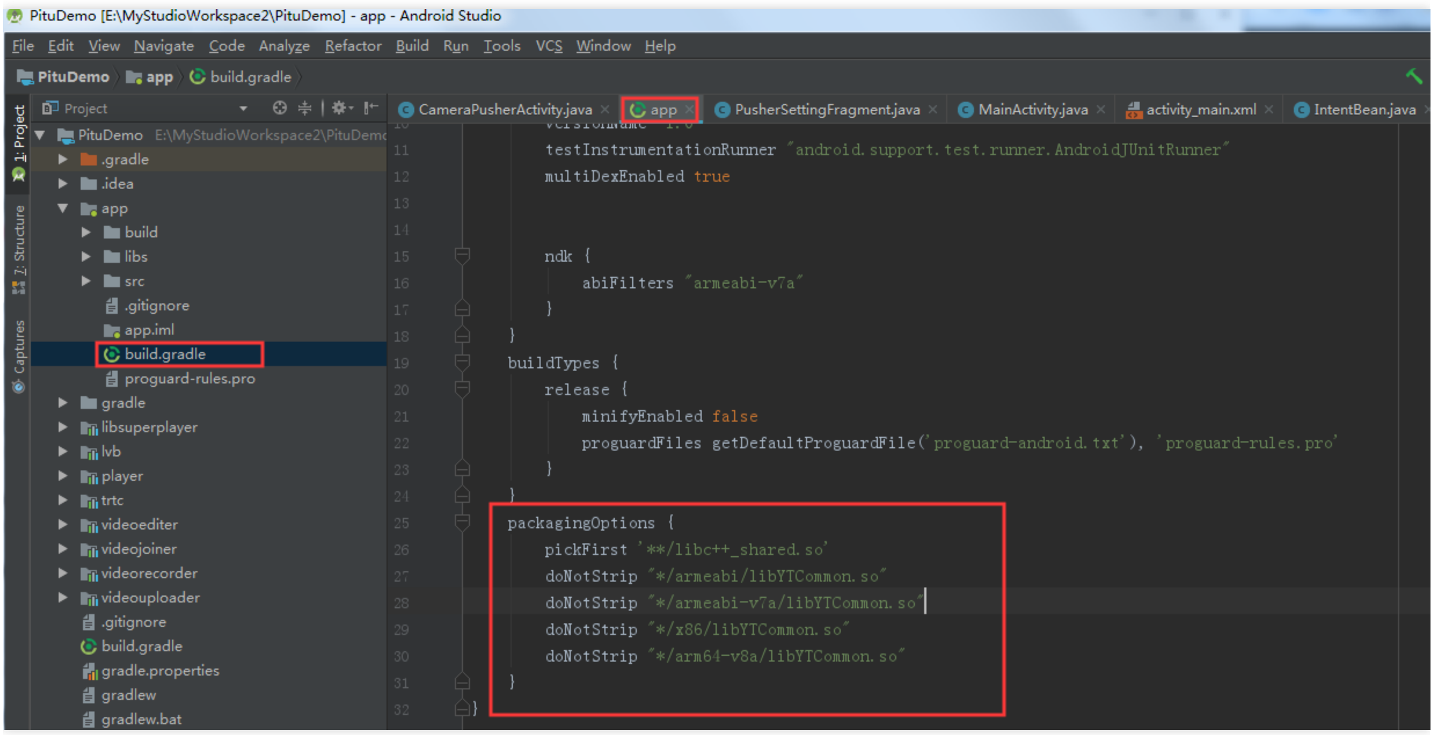
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.tencent.liteavsdkdemo"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        ndk {
            abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }

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

}

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

    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

6. In `defaultConfig` of `app/build.gradle` , specify the CPU architecture to be used by the application. Currently, LiteAVSDK supports armeabi, armeabi-v7a, and arm64-v8a.

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

7. Click **Sync Now** to complete the integration.

Setting Packaging Parameters



```
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        multiDexEnabled true


        ndk {
            abiFilters "armeabi-v7a"
        }
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    packagingOptions {
        pickFirst '**/libc++_shared.so'
        doNotStrip "*/armeabi/libYTCommon.so"
        doNotStrip "*/armeabi-v7a/libYTCommon.so"
        doNotStrip "*/x86/libYTCommon.so"
        doNotStrip "*/arm64-v8a/libYTCommon.so"
    }
}
```

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

Configuring Permissions

Configure permissions for your application in `AndroidManifest.xml`. LiteAVSDK needs 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" />
```

Configuring License

Log in to the CSS console, go to **MLVB SDK** > **License Management**, and click **Get License** to obtain a trial license. For detailed directions, see Applying for trial license. You will get two strings: a license URL and a decryption key.

Before you use LiteAVSDK features in your application, complete the following configurations (preferably in the application class).

```
public class MApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // The license URL obtained
String licenceKey = ""; // The license key obtained
TXLiveBase.getInstance().setLicence(this, licenceURL, licenceKey);
}
}
```

Configuring Obfuscation Rules

In the `proguard-rules.pro` file, add LiteAVSDK classes to the "do not obfuscate" list.

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

Features

Last updated：2022-09-14 17:26:49

We offer **MLVB Professional** for users outside the Chinese mainland. You need an professional license to unlock live streaming features such as publishing, playback, and basic filters (skin brightening and smoothing, etc.).

SDK and license

You can apply for a free trial license or purchase a license to use MLVB Professional.

Features

Module	Feature	Description	MLVB Professional
UI	Custom UI	Customizing UI. We provide a complete set of UI source code. You can use it directly or customize your own UI based on it.	✓
Publishing	RTMP	Stream publishing by hosts from mobile phones (live showroom)	✓
	Screen	Screen sharing by hosts from mobile phones (game streaming)	✓
Playback	RTMP	Playback over RTMP	✓
	FLV	Playback over HTTP + FLV	✓
	HLS	Playback over HLS (m3u8)	✓
	WebRTC	LEB	✓
Video on demand	Video on demand	Video playback on demand	✓
Mic connect	Host-audience interaction	One-to-many mic connect between the host and audience	✓
	Anchor competition	One-to-one competition between hosts	✓
Capturing and shooting	Aspect ratio	Multiple aspect ratios including 16:9, 4:3, and 1:1	✓
	Definition	SD, HD, and FHD video; custom bitrate, frame rate, and GOP	✓

Module	Feature	Description	MLVB Professional
	Capturing/Shooting control	Switching between the front and rear camera and lighting control	✓
	Watermark	Watermarking videos	✓
	Focus	Adjusting focal length	✓
	Focus mode	Manual or auto focus	✓
	Photo taking	Taking photos	✓
	Background music	Selecting a local MP3 file as the background music for capturing/shooting	✓
	Voice changing and reverb	Selecting a voice changing effect (e.g., little girl, middle-aged man) or reverb effect (e.g., karaoke room, hall) for capturing/shooting	✓
	Filters	Custom filters, which support strength adjusting	✓
	Basic retouching	Skin brightening, skin smoothing, and rosy skin, which support strength adjusting	✓
	Advanced filters	Eye enlarging, face slimming, chin slimming, chin adjustment, face shortening, and nose narrowing, which support strength adjusting	×
	Animated stickers	Identifying and reshaping facial features; adding stickers and widgets (provided as additional materials)	×
	AI-based keying	Identifying the foreground and replacing the background with other elements (provided as additional materials)	×
	Green screen keying	Removing the green parts of a video (for example, a green background) and replacing them with other elements	×

Support for RTC Publishing

Last updated：2022-06-14 12:41:46

In order to improve publishing performance under poor network conditions, we have added support for RTC publishing in addition to the traditional RTMP protocol. This document compares the performance of publishing under different network conditions using the two protocols. The video below shows the effects of watching streams published over different methods.

Note：
For instructions on how to publish streams, please see Publishing from Camera.

Performance Under Normal and Poor Network Conditions

Test method

We simulated different network conditions at the publishing end and observed the playback effects (the streams were played over CDNs, and network conditions at the playback end were normal).

Parameter configuration

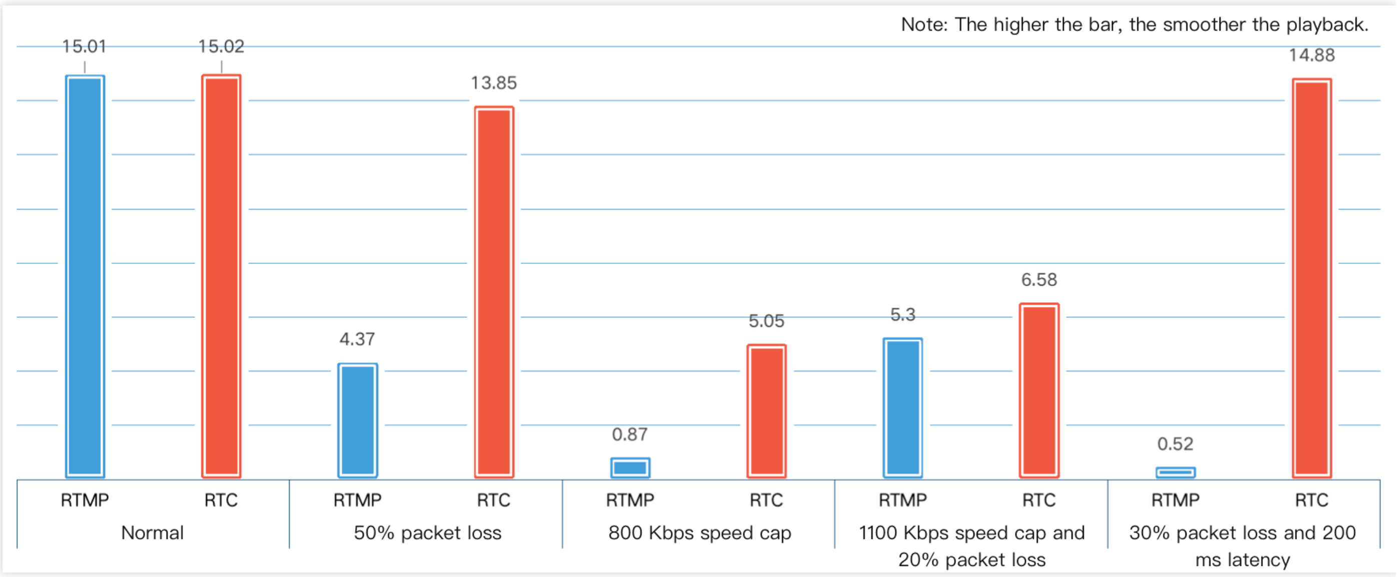
To prevent the use of different sources from affecting the results, we used V2TXLivePusher to publish the same video over RTMP and RTC.

Video parameters:

Parameter	Value
Resolution	720 x 1280
Bitrate	1800 Kbps
Frame rate	15

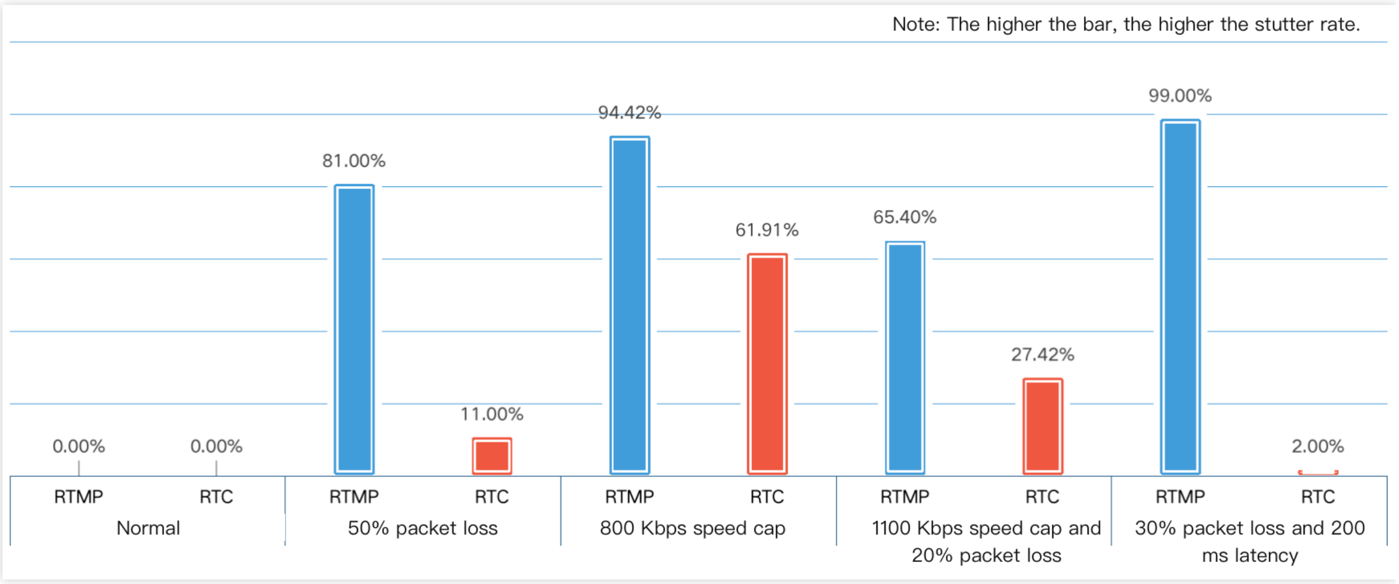
Comparison of performance under different network conditions

- **Frame rate**



Note：
For a description of the network parameters, see Appendix: Network Parameters.

- **Stutter rate**



Note：

For a description of the network parameters, see Appendix: Network Parameters.

Appendix: Network Parameters

Parameter	Description
Frame rate	Frames rendered per second
Packet loss	A packet loss rate of 50% means that for every 10 data packets sent, five fail to arrive at their destination.
Latency	A latency of 200 ms indicates that data packets are delivered by the network only 200 ms after they are sent by the SDK.
Transfer speed cap	A transfer speed cap of 800 Kbps means that 800 KB of data can be sent per second at most.
Stutter rate	Stutter occurs if the interval between the rendering of two consecutive frames exceeds 200 ms. Stutter rate is the total stuttering time divided by the total playback time.