Mobile Live Video Broadcasting Integration (No UI)

Product Documentation

Contents

Integration (No UI)

Running a Demo

iOS

Last updated：2024-01-13 15:49:41

This document describes how to quickly run Tencent Cloud MLVB-API-Example for iOS.

Environment Requirements

Xcode 9.0 or later

iPhone or iPad with iOS 9.0 or later

A valid developer signature for your project

Prerequisites

You have signed up for a Tencent Cloud account.

Directions

Step 1. Download the SDK and MLVB-API-Example source code

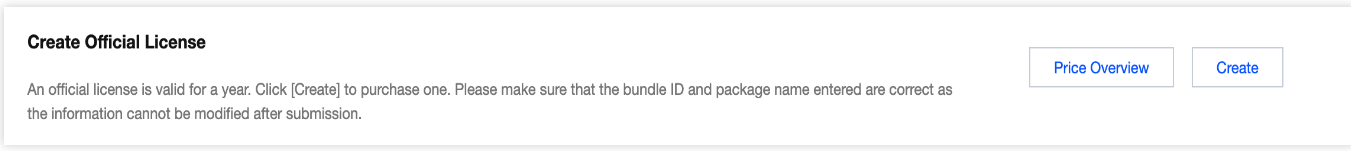
1. Download the package here as needed. Here, the Live Edition is used as an example.
2. Decompress the file after download.

Note

The source code can also be obtained from GitHub.

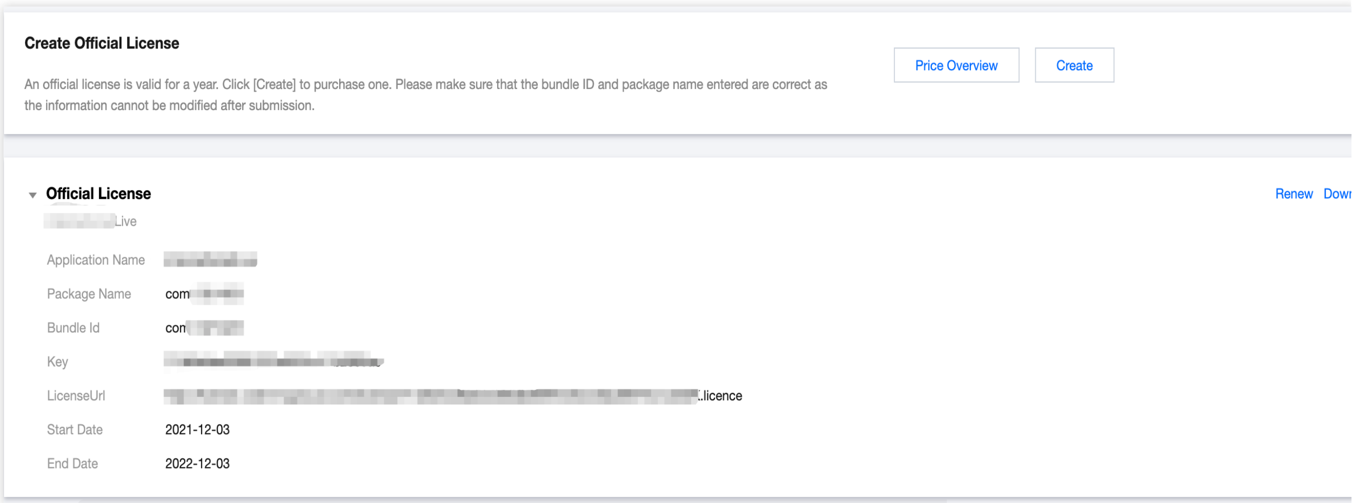
Step 2. Configure the license

1. Log in to the CSS console, select **MLVB SDK** > **License Management** on the left sidebar, and click **Create**.


Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

2. Enter the `App Name` , `Package Name` , and `Bundle ID` as needed, select the **Live streaming** feature module (**Live Push** + **Video Playback**), and click **Confirm**.
Package Name: Enter the **applicationId** in the **build.gradle** file in the `App` directory.
Bundle ID: Enter the **Bundle Identifier** of the project in **Xcode**.
3. After the free trial license is created successfully, the page will display the information of the generated license. **You need to pass in two parameters `Key` and `License URL` during initial SDK configuration. Store the following information properly:**



Create Official License

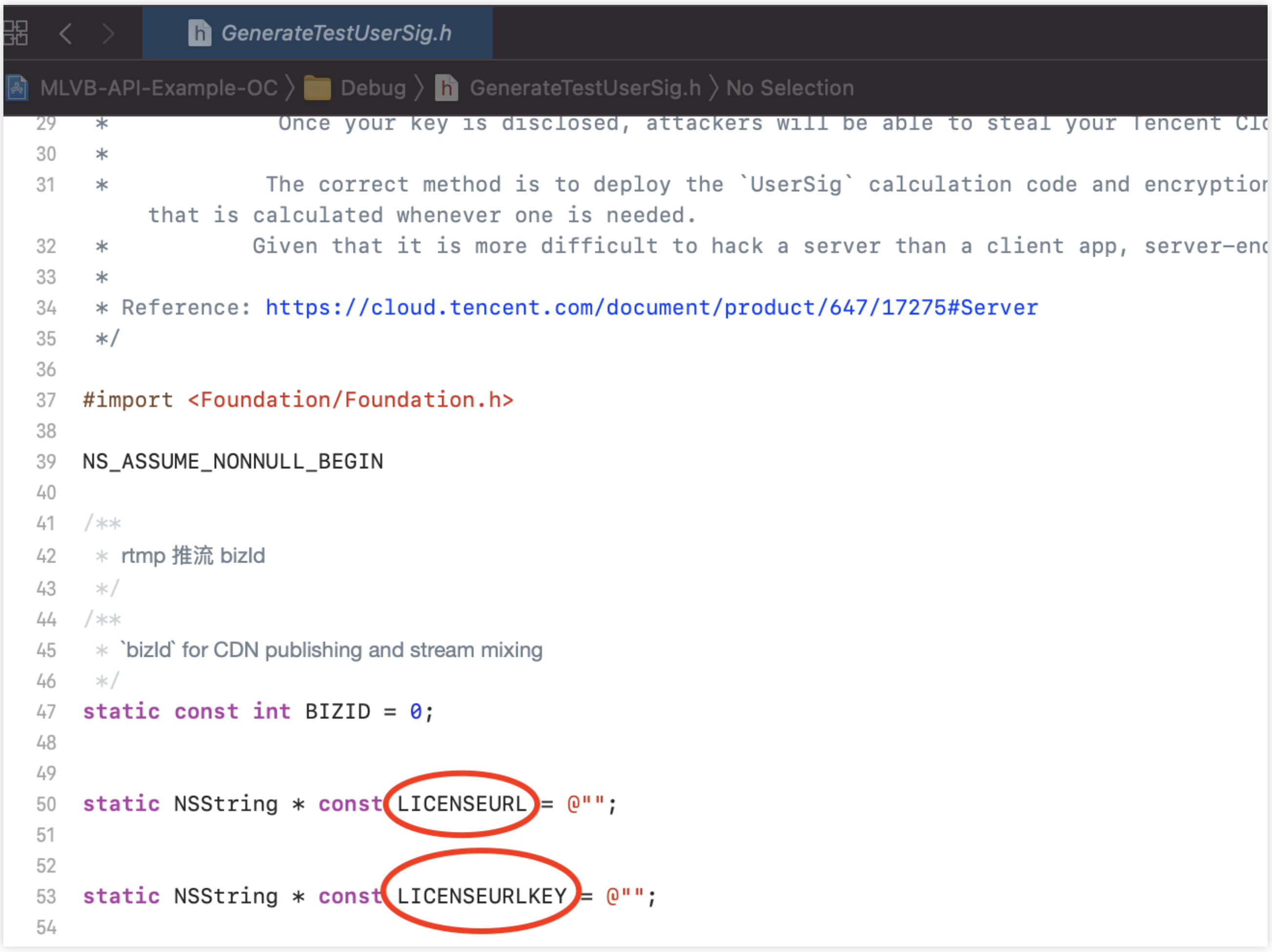
An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

[redacted]Live

Application Name	[redacted]
Package Name	com[redacted]
Bundle Id	con[redacted]
Key	[redacted]
LicenseUrl	[redacted].licence
Start Date	2021-12-03
End Date	2022-12-03

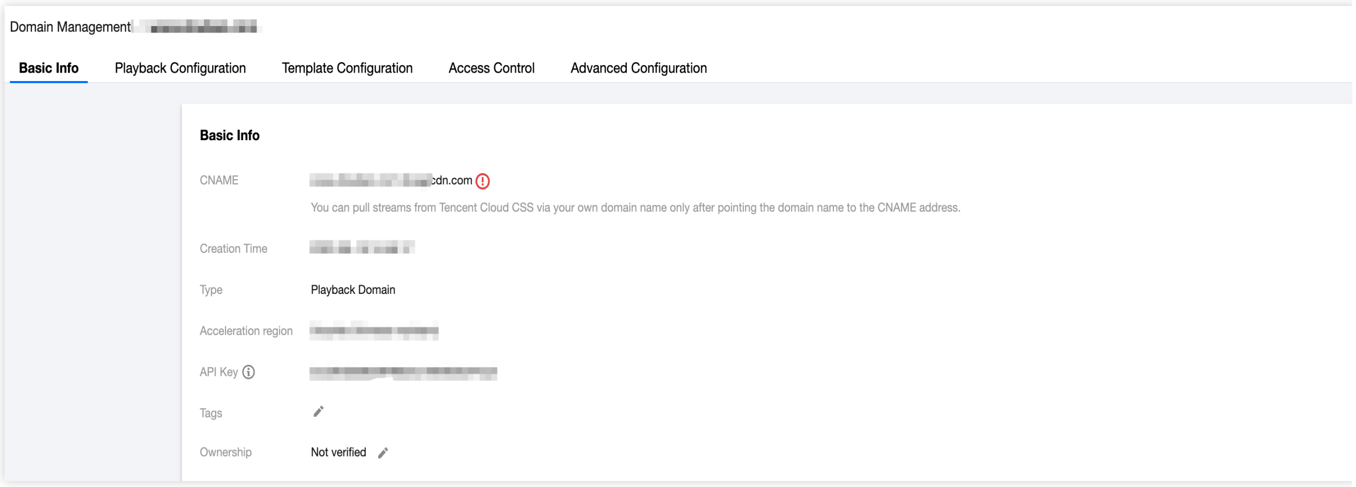
4. Open the `LiteAVSDK_Live_iOS_version number/MLVB-API-Example-OC/Debug/GenerateTestUserSig.h` file.
Set parameters in `GenerateTestUserSig.h` as follows:
LICENSEURL: Empty by default. Set it to the actual download license URL.
LICENSEURLKEY: Empty by default. Set it to the actual download license key.


GenerateTestUserSig.h
MLVB-API-Example-OC 〉 Debug 〉 GenerateTestUserSig.h 〉 No Selection

```
 *              Once your key is disclosed, attackers will be able to steal your Tencent Clo
 *
 *             The correct method is to deploy the `UserSig` calculation code and encryptior
    that is calculated whenever one is needed.
 *            Given that it is more difficult to hack a server than a client app, server-enc
 *
 * Reference: https://cloud.tencent.com/document/product/647/17275#Server
 */

#import <Foundation/Foundation.h>

NS_ASSUME_NONNULL_BEGIN

/**
 * rtmp 推流 bizId
 */
/**
 * `bizId` for CDN publishing and stream mixing
 */
static const int BIZID = 0;


static NSString * const LICENSEURL = @"";


static NSString * const LICENSEURLKEY = @"";

```

Step 3. Configure stream push/playback capabilities

1. Apply for a domain name in DNSPod and get an ICP filing for it.
2. Add the stream push/playback domain name in Domain Management in the **CSS console**. For detailed directions, see Adding Your Own Domain.
3. Configure the CNAME record for the domain name as instructed in Configuring CNAME.
4. After configuring the stream push/playback domain name, you can get the `CNAME` information on the **Basic Info** page of the domain name.



Domain Management [redacted]

Basic Info | Playback Configuration | Template Configuration | Access Control | Advanced Configuration

Basic Info

CNAME	[redacted]cdn.com You can pull streams from Tencent Cloud CSS via your own domain name only after pointing the domain name to the CNAME address.
Creation Time	[redacted]
Type	Playback Domain
Acceleration region	[redacted]
API Key	[redacted]
Tags	
Ownership	Not verified

5. Open the `LiteAVSDK_Live_iOS_version number/MLVB-API-Example-OC/Debug/GenerateTestUserSig.h` file.

Set parameters in `GenerateTestUserSig.h` as follows:

PUSH_DOMAIN: Set it to your stream push domain name.

PLAY_DOMAIN: Set it to your playback domain name.

LIVE_URL_KEY: This parameter is optional. It is used to generate authentication information such as `txSecret`. For more information on how to calculate it, see Publishing/Playback URL. You can query it in **Manage** > **Stream Push Configuration** > **Authentication Configuration** on the Domain Name page.

Configuring stream push parameters

1. Find and open the `LiteAVSDK_Live_iOS_version number/MLVB-API-Example-OC/Debug/GenerateTestUserSig.h` file.

2. Set parameters in the GenerateTestUserSig.h file based on the above service:

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

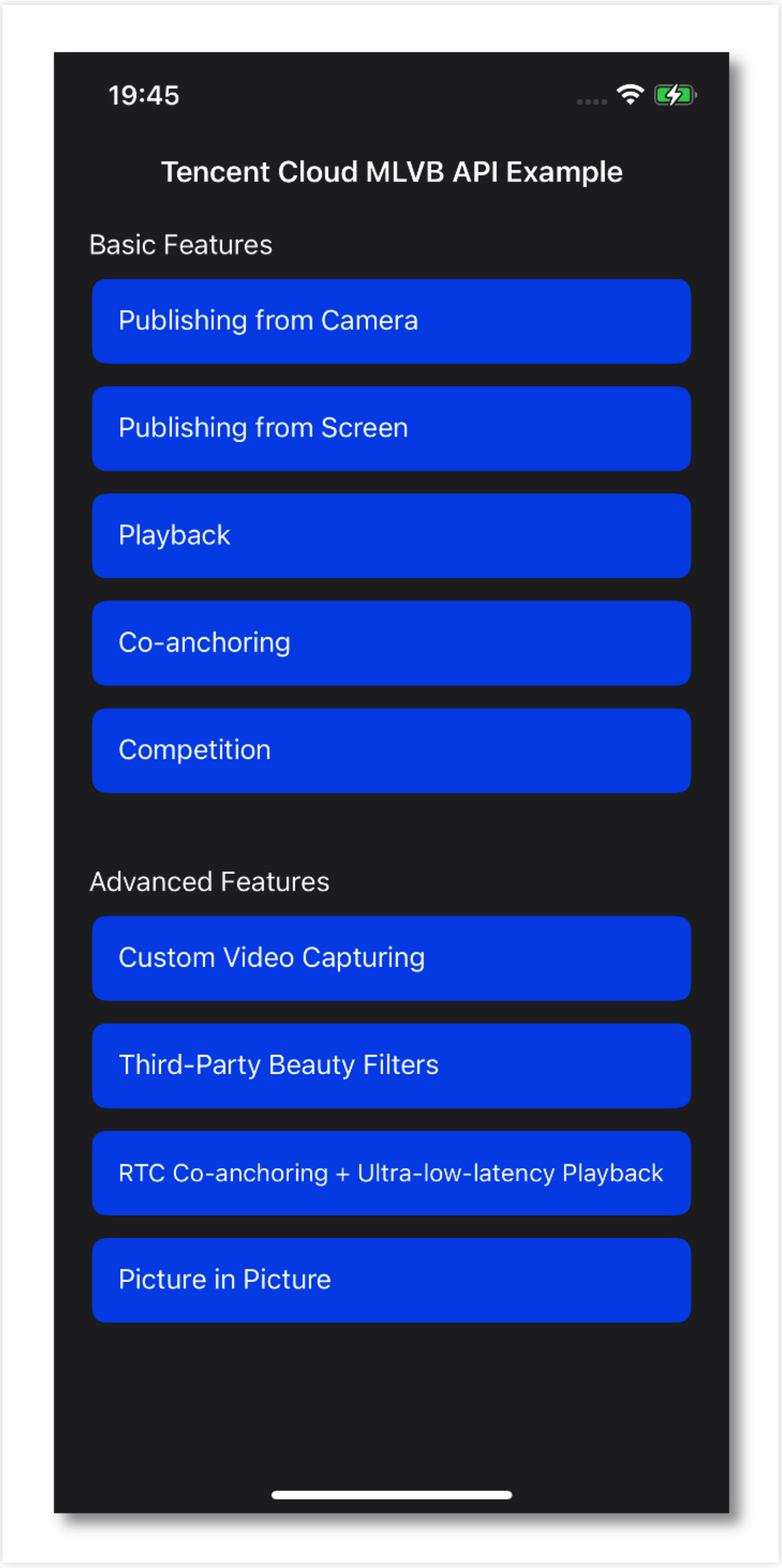
SECRETKEY: Empty by default. Set it to the actual secret key.

Stream push URL field description

You need to concatenate the specific stream push/pull URL string based on the used protocol as instructed in Publishing/Playback URL. A string has been concatenated in the demo, and the stream can be played back after you run the demo.

Step 5. Compile and run

Open the demo project `MLVB-API-Example-OC` with Xcode 9.0 or later and click **Run**.


19:45
Tencent Cloud MLVB API Example
Basic Features
Publishing from Camera
Publishing from Screen
Playback
Co-anchoring
Competition
Advanced Features
Custom Video Capturing
Third-Party Beauty Filters
RTC Co-anchoring + Ultra-low-latency Playback
Picture in Picture

Android

Last updated：2024-01-13 15:49:41

This document describes how to quickly run the Tencent Cloud MLVB-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 later
Device on Android 4.1 or above for the application

Prerequisites

You have signed up for a Tencent Cloud account.

Directions

Step 1. Download the SDK and MLVB-API-Example source code

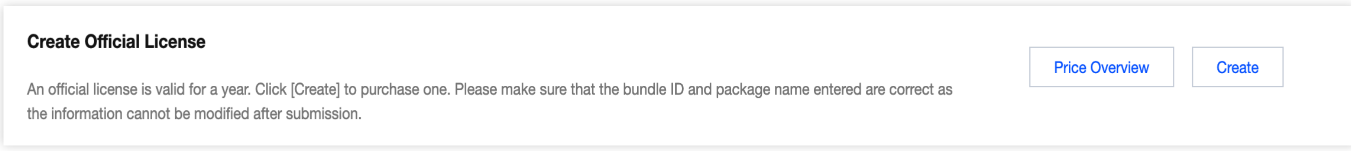
1. Download the package here as needed. Here, the Live Edition is used as an example.
2. Decompress the file after download.

Note

The source code can also be obtained from GitHub.

Step 2. Configure the license

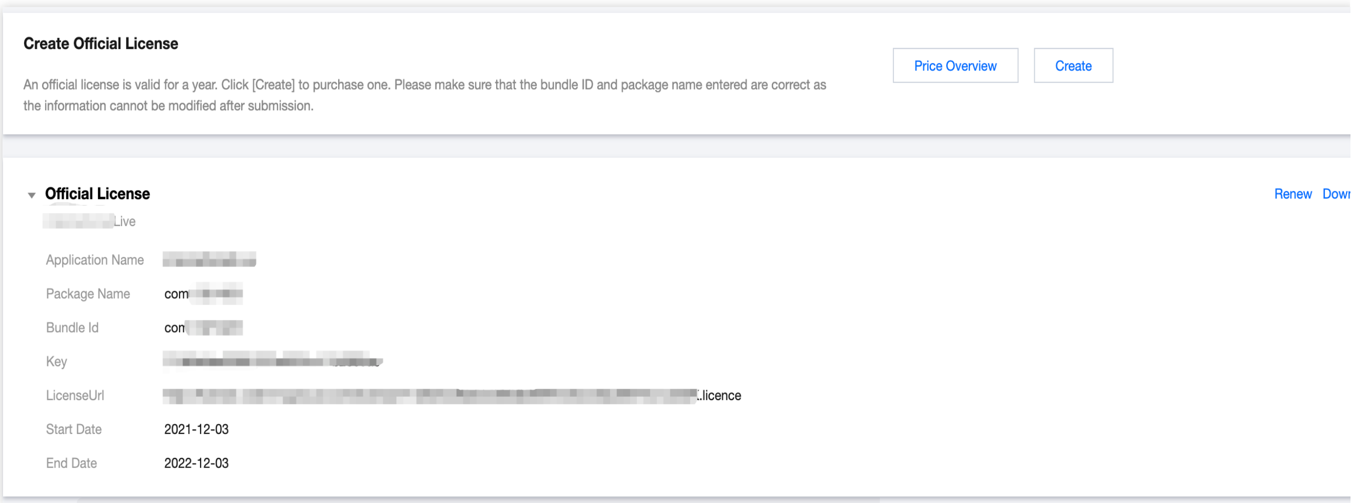
1. Log in to the CSS console, select **MLVB SDK** > **License Management** on the left sidebar, and click **Create**.


Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

2. Enter the `App Name` , `Package Name` , and `Bundle ID` as needed and click **Confirm**.
Package Name: Enter the **applicationId** in the **build.gradle** file in the `App` directory.
Bundle ID: Enter the **Bundle Identifier** of the project in **Xcode**.

3. After the license is created successfully, the page will display the information of the generated license. **You need to pass in two parameters, `Key` and `License URL`, during initial SDK configuration. Store the following information properly.**


Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

Live

Application Name	[illegible]
Package Name	com
Bundle Id	con
Key	[illegible]
LicenseUrl	.licence
Start Date	2021-12-03
End Date	2022-12-03

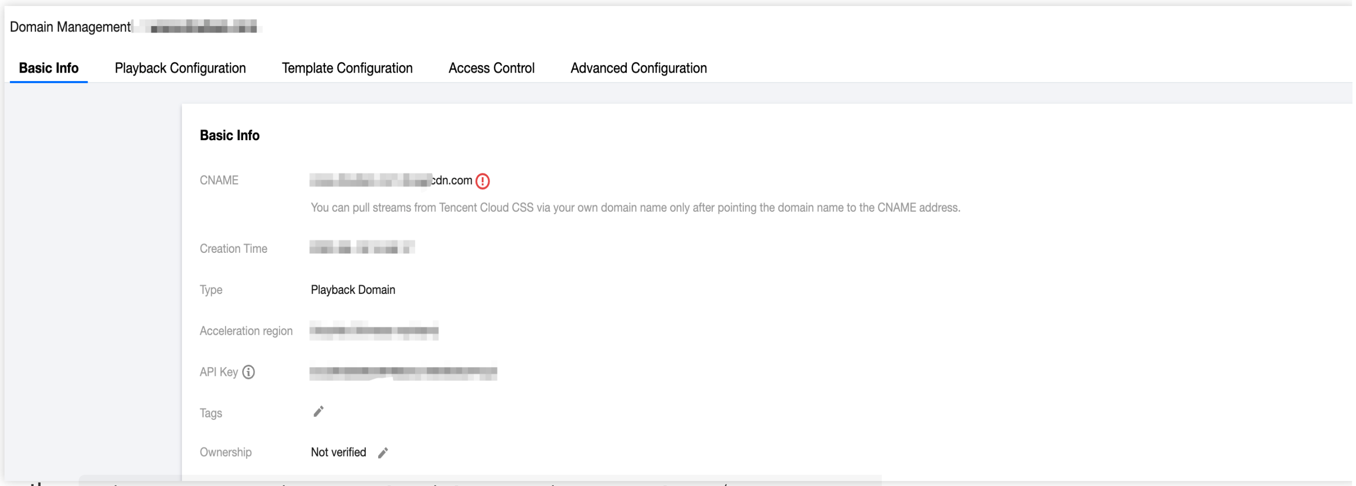
4. Open the `LiteAVSDK_Live_Android_version number/MLVB-API-Example/Debug/src/main/java/com/tencent/mlvb/debug/GenerateTestUserSig.java` file. Set parameters in `GenerateTestUserSig.java` as follows:

`LICENSEURL`: A placeholder by default. Set it to the actual download license URL.

LICENSEURLKEY: A placeholder by default. Set it to the actual download license key.

Step 3. Configure stream push/playback capabilities

1. Apply for a domain name in DNSPod and get an ICP filing for it.
2. Add the stream push/playback domain name in Domain Management in the **CSS console**. For detailed directions, see Adding Your Own Domain.
3. Configure the CNAME record for the domain name as instructed in Configuring CNAME.
4. After configuring the stream push/playback domain name, you can get the `CNAME` information on the **Basic Info** page of the domain name.


Domain Management

Basic Info | Playback Configuration | Template Configuration | Access Control | Advanced Configuration

Basic Info

CNAME: cdn.com
You can pull streams from Tencent Cloud CSS via your own domain name only after pointing the domain name to the CNAME address.

Creation Time:

Type: Playback Domain

Acceleration region:

API Key:

Tags:

Ownership: Not verified

5. Open the `LiteAVSDK_Live_Android_version number/MLVB-API-Example/Debug/src/main/java/com/tencent/mlvb/debug/GenerateTestUserSig.java` file.
Set parameters in `GenerateTestUserSig.java` as follows:
PUSH_DOMAIN: Set it to your stream push domain name.
PLAY_DOMAIN: Set it to your playback domain name.
LIVE_URL_KEY: This parameter is optional. It is used to generate authentication information such as `txSecret` . For more information on how to calculate it, see Publishing/Playback URL. You can query it in **Manage** > **Stream Push Configuration** > **Authentication Configuration** on the Domain Name page.

Configuring stream push parameters

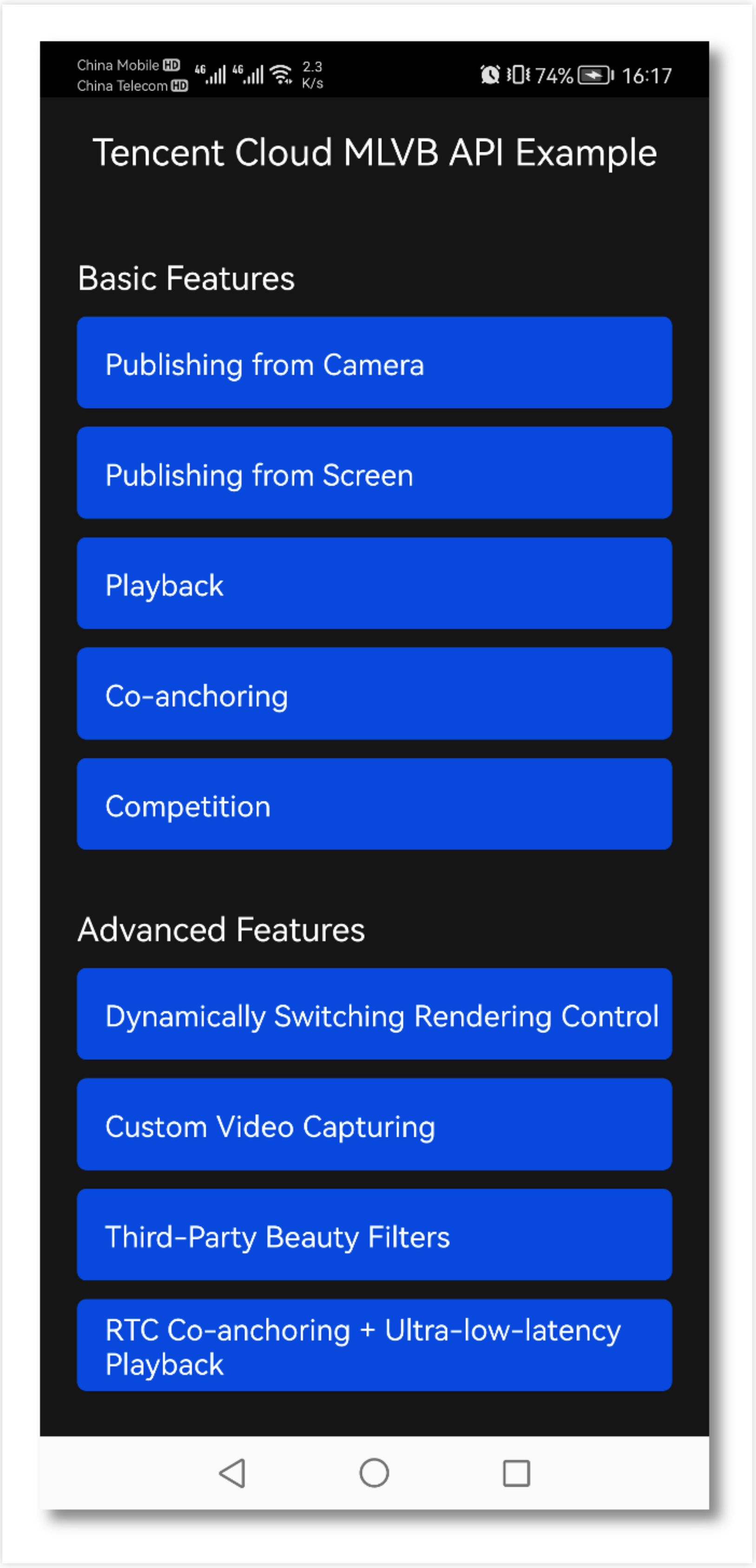
1. Find and open the `LiteAVSDK_Live_Android version number/MLVB-API-Example/Debug/src/main/java/com/tencent/mlvb/debug/GenerateTestUserSig.java` file.
2. Set parameters in the GenerateTestUserSig.java file based on the above service:
SDKAPPID: A placeholder by default. Set it to the actual `SDKAppID` .
SECRETKEY: A placeholder by default. Set it to the actual secret key.

Stream push URL field description

You need to concatenate the specific stream push/pull URL string based on the used protocol as instructed in Publishing/Playback URL. A string has been concatenated in the demo, and the stream can be played back after you run the demo.

Step 5. Compile and run

Open the demo project `MLVB-API-Example` with Android Studio 3.5 or later and click **Run**.



Tencent Cloud MLVB API Example

Basic Features

Publishing from Camera

Publishing from Screen

Playback

Co-anchoring

Competition

Advanced Features

Dynamically Switching Rendering Control

Custom Video Capturing

Third-Party Beauty Filters

RTC Co-anchoring + Ultra-low-latency Playback

SDK Integration
iOS

Last updated：2024-01-13 15:49:41

This document describes how to quickly integrate the MLVB SDK (iOS) of Tencent Video Cloud Toolkit into your project. The directions below use the full-featured Live Edition as an example.

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

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

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

target 'App' do
pod 'TXLiteAVSDK_Live'
end
```

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
```

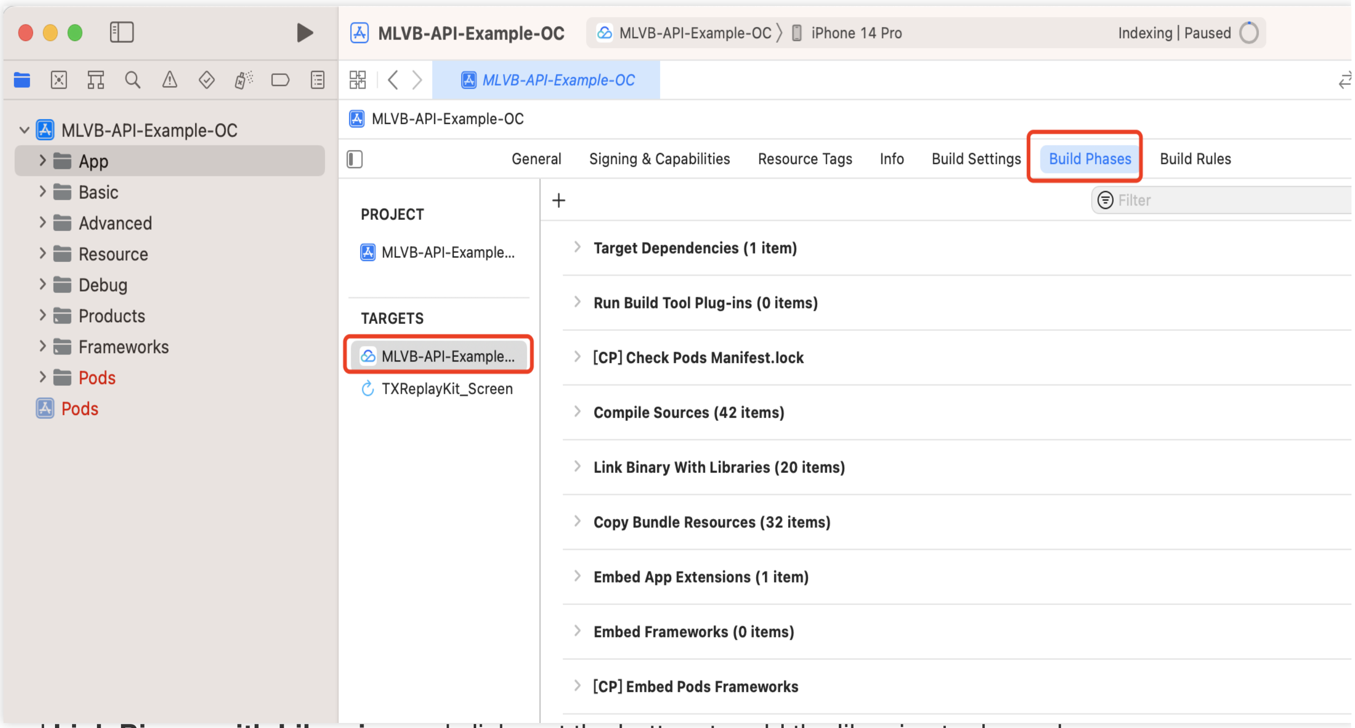
Or, run the following command to update the local repository:

```
pod update
```

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

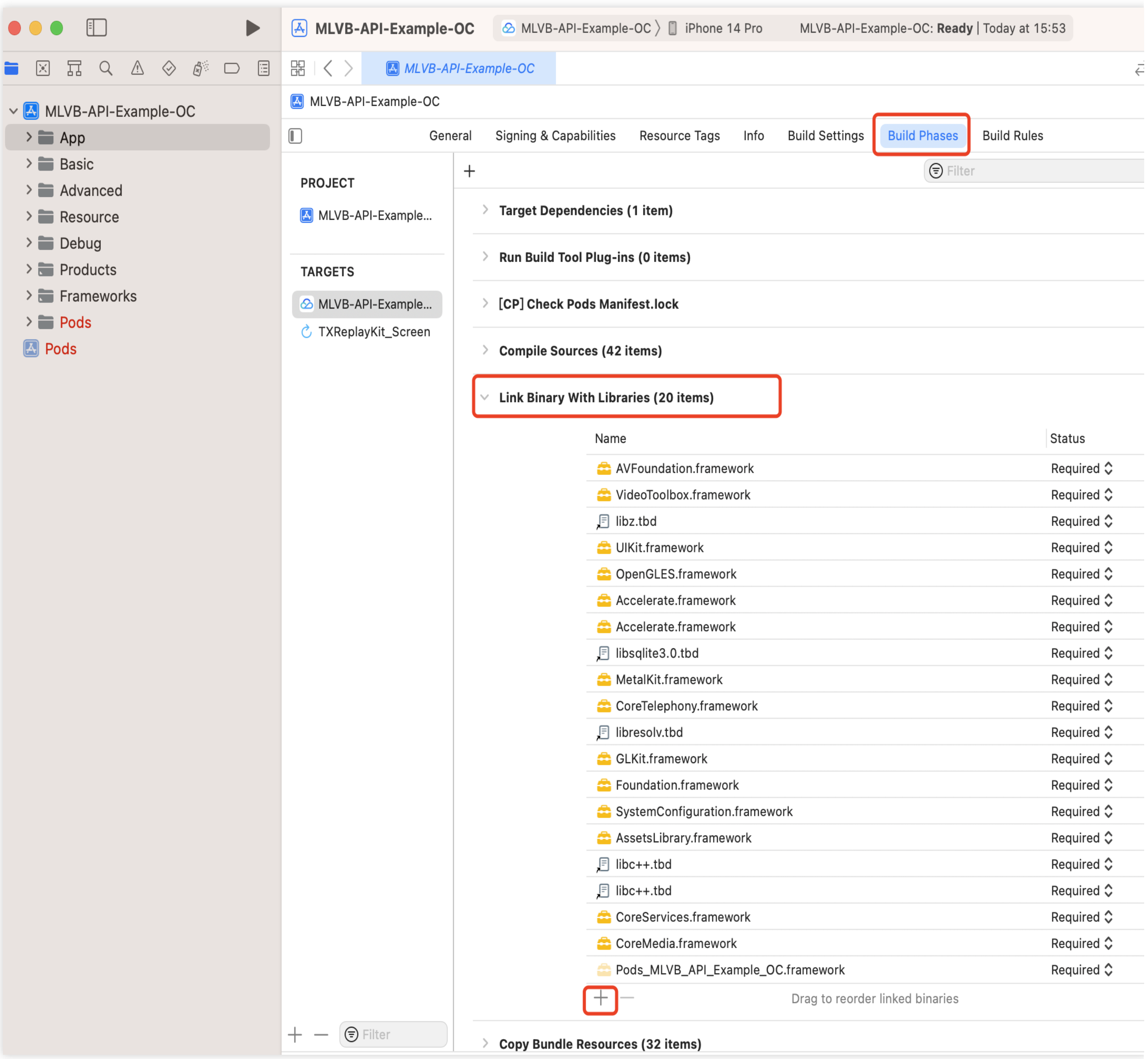
Manual integration

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



Target Dependencies (1 item)
Run Build Tool Plug-ins (0 items)
[CP] Check Pods Manifest.lock
Compile Sources (42 items)
Link Binary With Libraries (20 items)
Copy Bundle Resources (32 items)
Embed App Extensions (1 item)
Embed Frameworks (0 items)
[CP] Embed Pods Frameworks

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

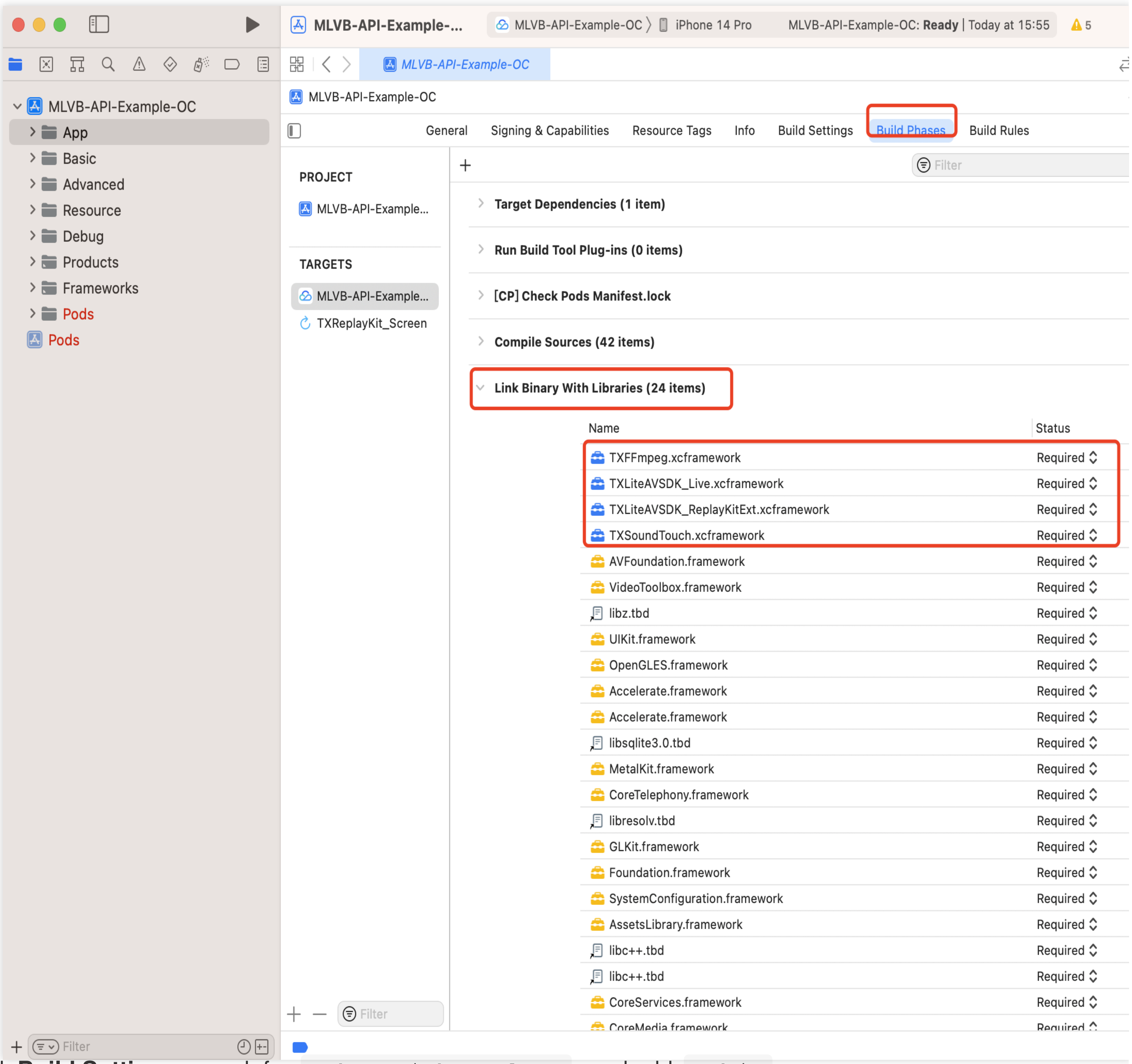

MLVB-API-Example-OC
MLVB-API-Example-OC 〉 iPhone 14 Pro
MLVB-API-Example-OC: Ready | Today at 15:53
App
Basic
Advanced
Resource
Debug
Products
Frameworks
Pods
Pods
PROJECT
MLVB-API-Example...
TARGETS
MLVB-API-Example...
TXReplayKit_Screen
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
Filter
Target Dependencies (1 item)
Run Build Tool Plug-ins (0 items)
[CP] Check Pods Manifest.lock
Compile Sources (42 items)
Link Binary With Libraries (20 items)
Drag to reorder linked binaries
Copy Bundle Resources (32 items)

Name	Status
AVFoundation.framework	Required
VideoToolbox.framework	Required
libz.tbd	Required
UIKit.framework	Required
OpenGLES.framework	Required
Accelerate.framework	Required
Accelerate.framework	Required
libsqlite3.0.tbd	Required
MetalKit.framework	Required
CoreTelephony.framework	Required
libresolv.tbd	Required
GLKit.framework	Required
Foundation.framework	Required
SystemConfiguration.framework	Required
AssetsLibrary.framework	Required
libc++.tbd	Required
libc++.tbd	Required
CoreServices.framework	Required
CoreMedia.framework	Required
Pods_MLVB_API_Example_OC.framework	Required

4. Add the downloaded `TXLiteAVSDK_Live.xcframework` 、 `TXFFmpeg.xcframework` 、 `TXSoundTouch.xcframework` and the libraries it depends on.

```
AVFoundation.framework
VideoToolbox.framework
libz.tbd
OpenGLES.framework
Accelerate.framework
libsqlite3.0.tbd
MetalKit.framework
CoreTelephony.framework
libresolv.tbd
GLKit.framework
```

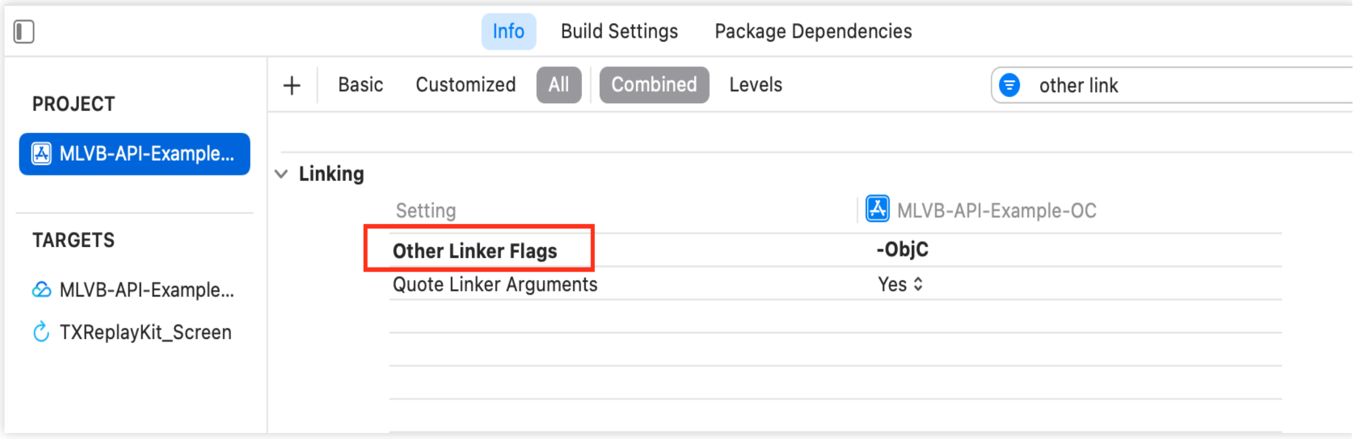
```
Foundation.framework
SystemConfiguration.framework
AssetsLibrary.framework
libc++.tbd
CoreServices.framework
CoreMedia.framework
```



Target Dependencies (1 item)
Run Build Tool Plug-ins (0 items)
[CP] Check Pods Manifest.lock
Compile Sources (42 items)
Link Binary With Libraries (24 items)

Name	Status
TXFFmpeg.xcframework	Required
TXLiteAVSDK_Live.xcframework	Required
TXLiteAVSDK_ReplayKitExt.xcframework	Required
TXSoundTouch.xcframework	Required
AVFoundation.framework	Required
VideoToolbox.framework	Required
libz.tbd	Required
UIKit.framework	Required
OpenGLES.framework	Required
Accelerate.framework	Required
Accelerate.framework	Required
libsqlite3.0.tbd	Required
MetalKit.framework	Required
CoreTelephony.framework	Required
libresolv.tbd	Required
GLKit.framework	Required
Foundation.framework	Required
SystemConfiguration.framework	Required
AssetsLibrary.framework	Required
libc++.tbd	Required
libc++.tbd	Required
CoreServices.framework	Required
CoreMedia.framework	Required

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



Linking

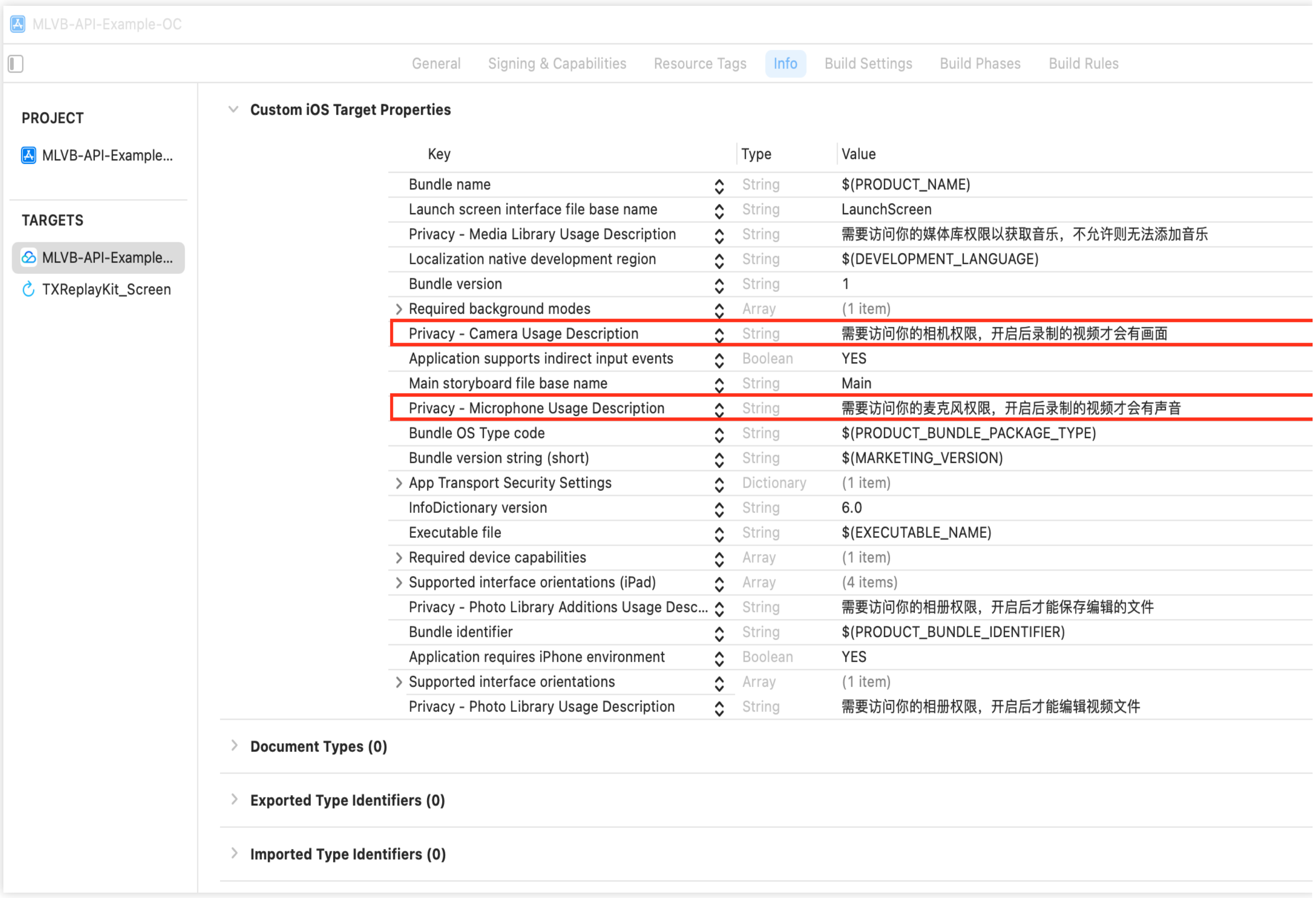
Setting	MLVB-API-Example-OC
Other Linker Flags	-ObjC
Quote Linker Arguments	Yes

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



MLVB-API-Example-OC

General | Signing & Capabilities | Resource Tags | Info | Build Settings | Build Phases | Build Rules

PROJECT
MLVB-API-Example...

TARGETS
MLVB-API-Example...
TXReplayKit_Screen

Custom iOS Target Properties

Key	Type	Value
Bundle name	String	$(PRODUCT_NAME)
Launch screen interface file base name	String	LaunchScreen
Privacy - Media Library Usage Description	String	需要访问你的媒体库权限以获取音乐，不允许则无法添加音乐
Localization native development region	String	$(DEVELOPMENT_LANGUAGE)
Bundle version	String	1
Required background modes	Array	(1 item)
Privacy - Camera Usage Description	String	需要访问你的相机权限，开启后录制的视频才会有画面
Application supports indirect input events	Boolean	YES
Main storyboard file base name	String	Main
Privacy - Microphone Usage Description	String	需要访问你的麦克风权限，开启后录制的视频才会有声音
Bundle OS Type code	String	$(PRODUCT_BUNDLE_PACKAGE_TYPE)
Bundle version string (short)	String	$(MARKETING_VERSION)
App Transport Security Settings	Dictionary	(1 item)
InfoDictionary version	String	6.0
Executable file	String	$(EXECUTABLE_NAME)
Required device capabilities	Array	(1 item)
Supported interface orientations (iPad)	Array	(4 items)
Privacy - Photo Library Additions Usage Desc...	String	需要访问你的相册权限，开启后才能保存编辑的文件
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
Application requires iPhone environment	Boolean	YES
Supported interface orientations	Array	(1 item)
Privacy - Photo Library Usage Description	String	需要访问你的相册权限，开启后才能编辑视频文件

Document Types (0)

Exported Type Identifiers (0)

Imported Type Identifiers (0)

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_Live;
```

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

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

Configuring License

Click Get License to obtain a 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:]`):

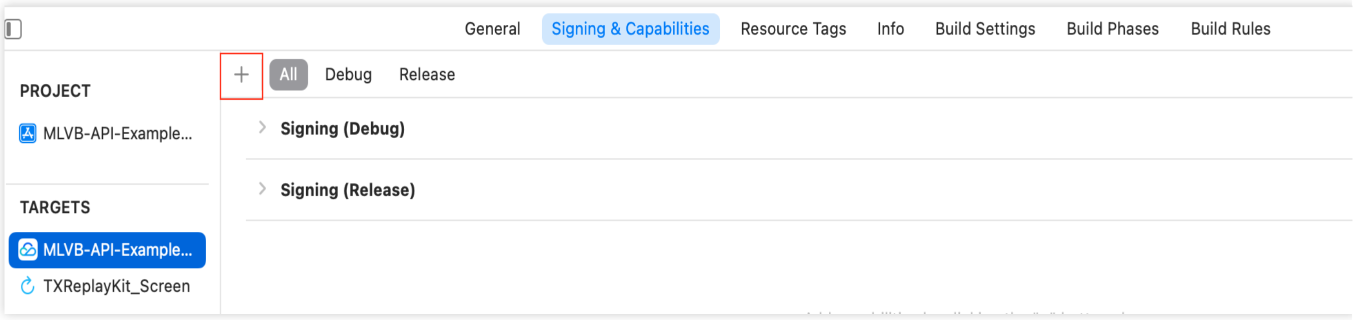
```
@import TXLiteAVSDK_Live;
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<The license URL obtained>";
    NSString * const licenceKey = @"<The key obtained>";

    //V2TXLivePremier can be found in the "V2TXLivePremier.h" header file
    [V2TXLivePremier setEnvironment:@"GDPR"];
    [V2TXLivePremier setLicence:licenceURL key:licenceKey];
    [V2TXLivePremier setObserver:self];
    NSLog(@"SDK Version = %@", [V2TXLivePremier getSDKVersionStr]);
}
@end
```

FAQs

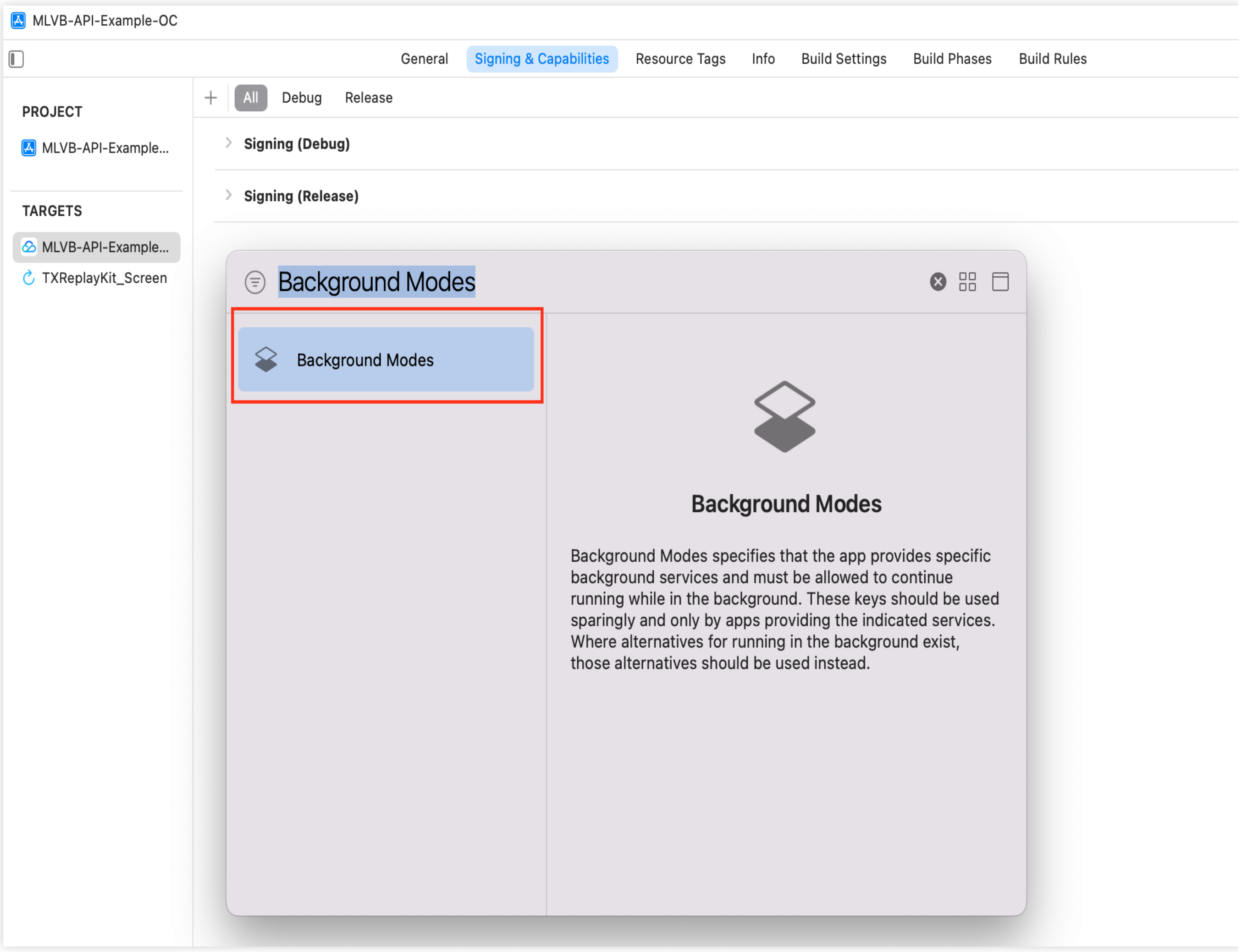
1. Can I run LiteAVSDK in the background?

Yes, you can. If you want the SDK to run in the background, the operation is as follows:

1. Select the current project, select **Signing&Capabilities** , click **+** in the upper left corner, as shown in the figure:

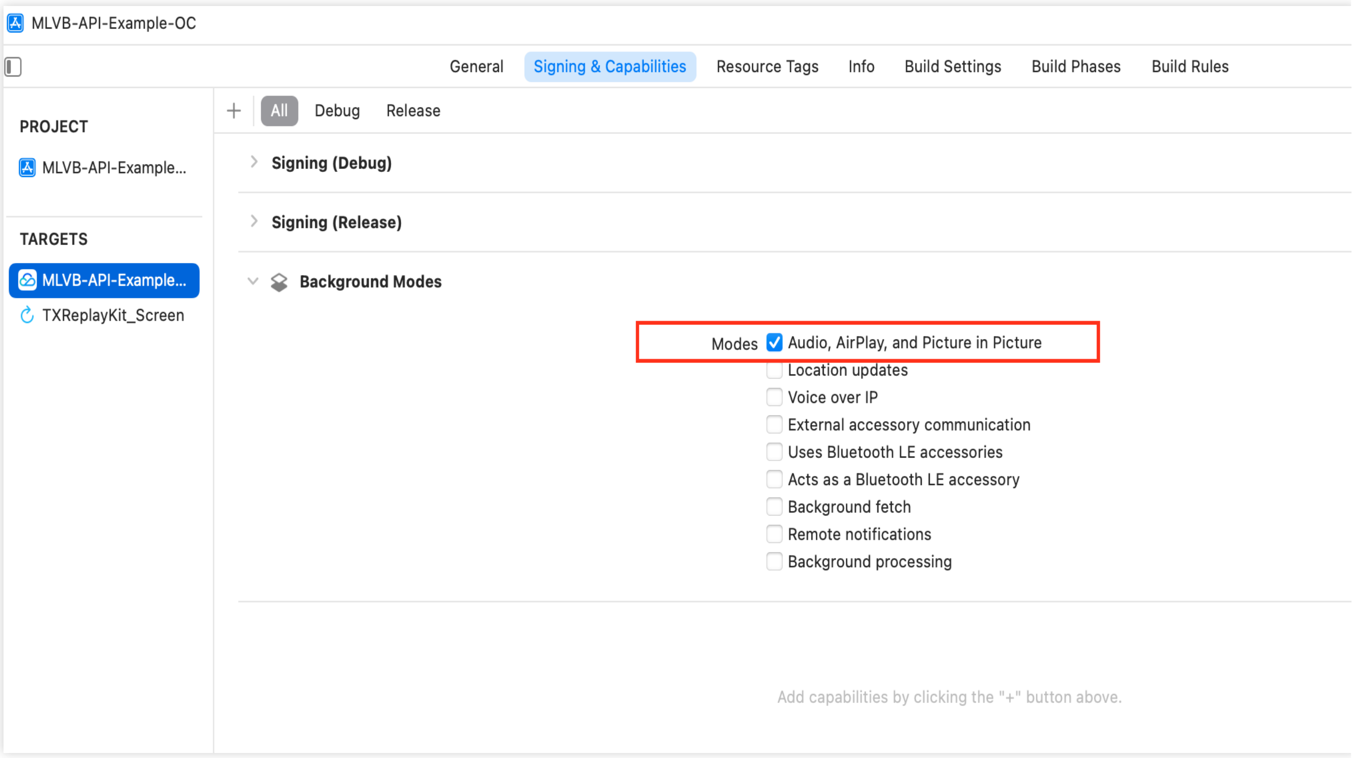


2. Select **Background Modes**.



Background Modes specifies that the app provides specific background services and must be allowed to continue running while in the background. These keys should be used sparingly and only by apps providing the indicated services. Where alternatives for running in the background exist, those alternatives should be used instead.

3. Check **Audio, AirPlay and Picture in Picture** in **Background Modes**, as shown below:



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

2. How to solve the problem of symbol conflict between multiple SDKs of LiteAVSDK series such as live SDK/real-time audio/video/player integrated in the project?

If you integrate 2 or more LiteAVSDK products (live broadcast, player, TRTC, short video), there will be a library conflict problem when compiling, because some SDK underlying libraries have the same symbol files, it is recommended to integrate only one full-featured version of the SDK. Live broadcast, player, TRTC, and short video are all included in one SDK.For details, please refer to SDK Download.

Android

Last updated：2024-01-13 15:49:41

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

Environment Requirements

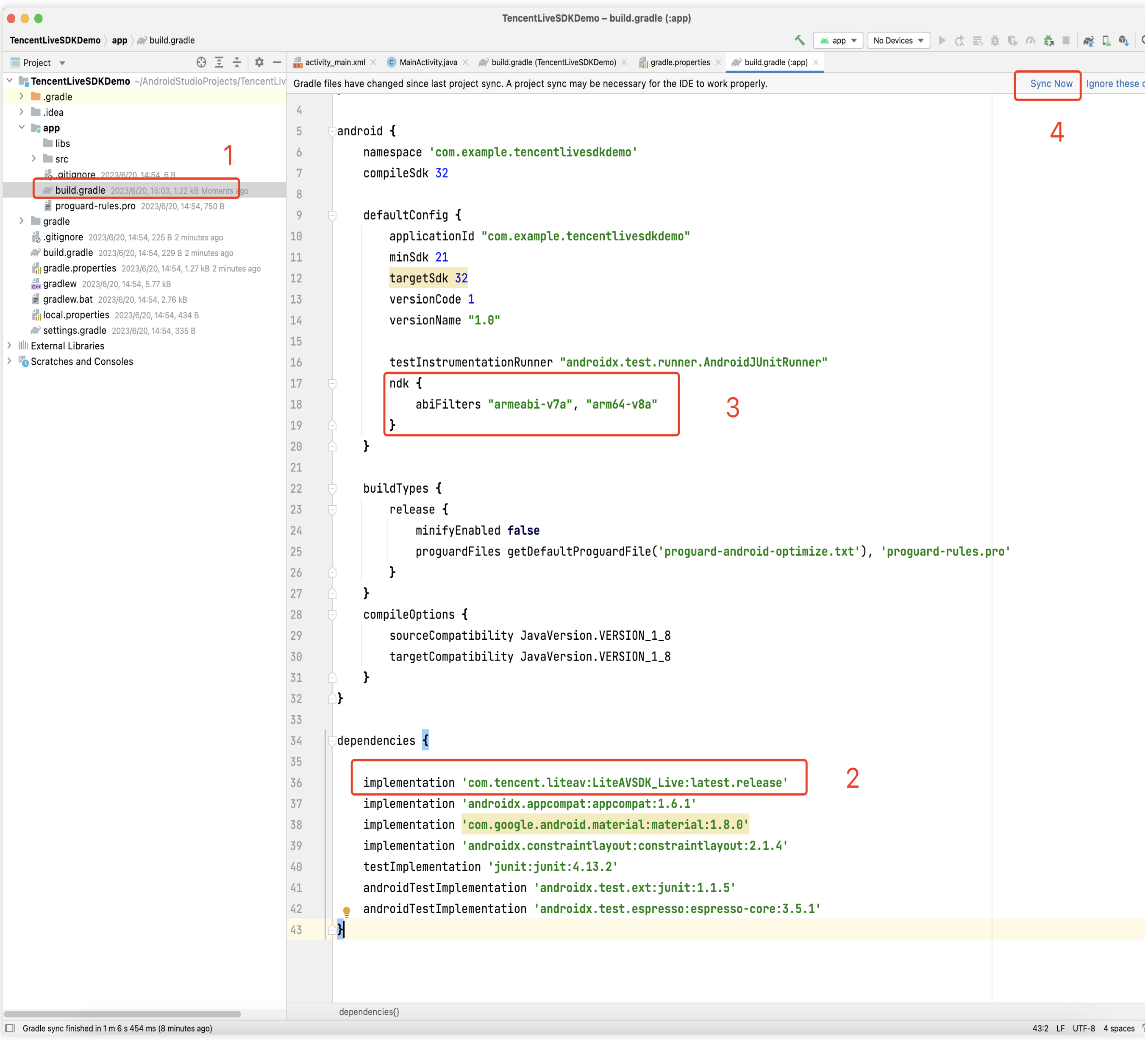
Android Studio 3.5 or above
Android 4.1 (SDK API level 16) or above

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.


TencentLiveSDKDemo – build.gradle (:app)
TencentLiveSDKDemo 〉 app 〉 build.gradle
app
No Devices
Project
activity_main.xml
MainActivity.java
build.gradle (TencentLiveSDKDemo)
gradle.properties
build.gradle (:app)
Sync Now
Ignore these
TencentLiveSDKDemo ~/AndroidStudioProjects/TencentLiv
.gradle
.idea
app
libs
src
.gitignore 2023/6/20, 14:54, 6 B
build.gradle 2023/6/20, 15:03, 1.22 kB Moments ago
proguard-rules.pro 2023/6/20, 14:54, 750 B
gradle
.gitignore 2023/6/20, 14:54, 225 B 2 minutes ago
build.gradle 2023/6/20, 14:54, 229 B 2 minutes ago
gradle.properties 2023/6/20, 14:54, 1.27 kB 2 minutes ago
gradlew 2023/6/20, 14:54, 5.77 kB
gradlew.bat 2023/6/20, 14:54, 2.76 kB
local.properties 2023/6/20, 14:54, 434 B
settings.gradle 2023/6/20, 14:54, 335 B
External Libraries
Scratches and Consoles
1
2
3
4
dependencies{}
43:2 LF UTF-8 4 spaces

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

```
android {
    namespace 'com.example.tencentlivesdkdemo'
    compileSdk 32

    defaultConfig {
        applicationId "com.example.tencentlivesdkdemo"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        ndk {
            abiFilters "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_Live:latest.release'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Gradle sync finished in 1 m 6 s 454 ms (8 minutes ago)

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

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

Or

```
dependencies {
    implementation 'com.tencent.liteav:LiteAVSDK_Live: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-v7a", "arm64-v8a"
    }
}
```

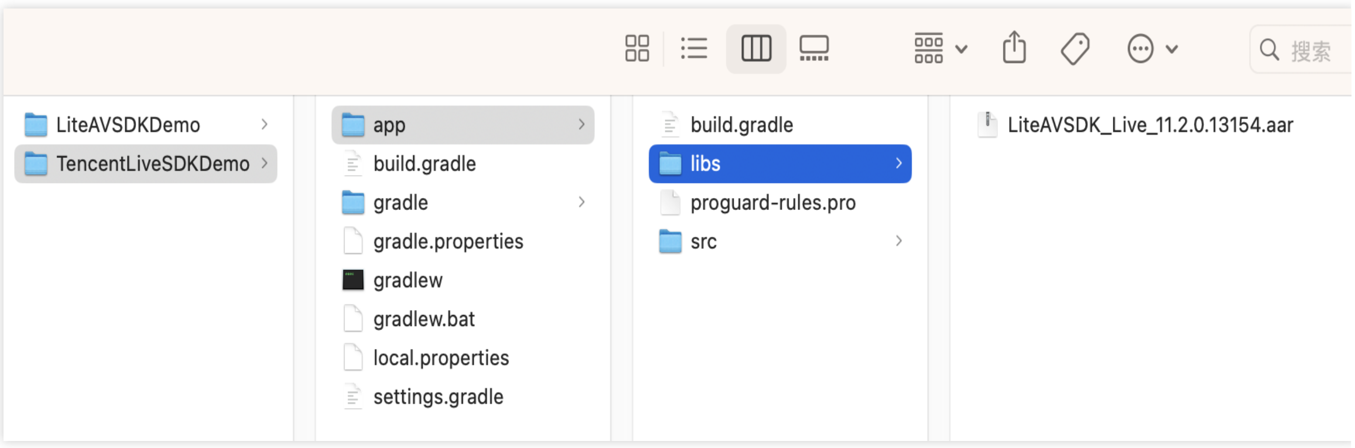
4. Click the **Sync Now** button

to sync the SDK. If you have no problem accessing Maven Central, the SDK will be downloaded and integrated into your project automatically.

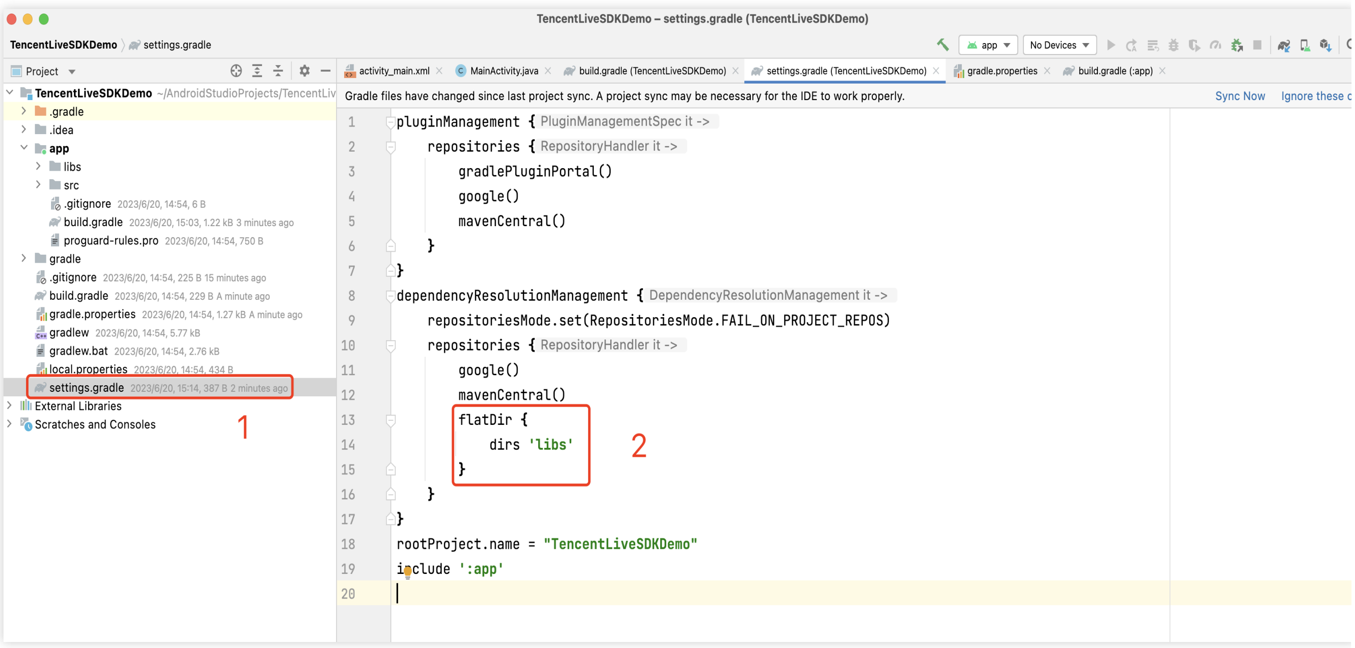
Method 2: manual download (AAR)

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

1. Download LiveAVSDK and decompress the file.
2. Copy the AAR file in the SDK directory to the **app/libs** directory of your project.



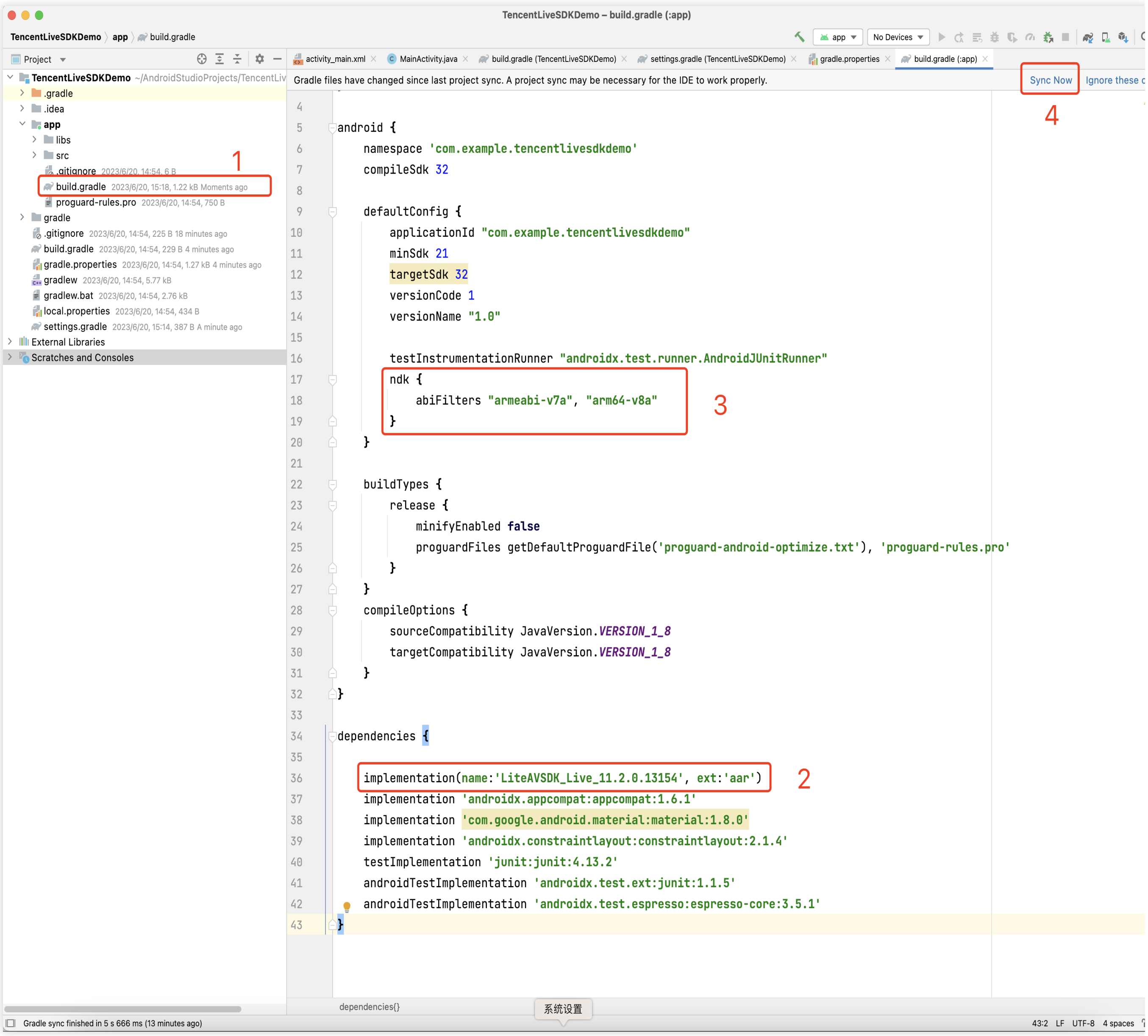
3. Add **flatDir** to `build.gradle` under your project's root directory to specify the local path for the repository.


TencentLiveSDKDemo – settings.gradle (TencentLiveSDKDemo)
TencentLiveSDKDemo 〉 settings.gradle
app
No Devices
Project
activity_main.xml
MainActivity.java
build.gradle (TencentLiveSDKDemo)
settings.gradle (TencentLiveSDKDemo)
gradle.properties
build.gradle (:app)
TencentLiveSDKDemo ~/AndroidStudioProjects/TencentLiv
.gradle
.idea
app
libs
src
.gitignore 2023/6/20, 14:54, 6 B
build.gradle 2023/6/20, 15:03, 1.22 kB 3 minutes ago
proguard-rules.pro 2023/6/20, 14:54, 750 B
gradle
.gitignore 2023/6/20, 14:54, 225 B 15 minutes ago
build.gradle 2023/6/20, 14:54, 229 B A minute ago
gradle.properties 2023/6/20, 14:54, 1.27 kB A minute ago
gradlew 2023/6/20, 14:54, 5.77 kB
gradlew.bat 2023/6/20, 14:54, 2.76 kB
local.properties 2023/6/20, 14:54, 434 B
settings.gradle 2023/6/20, 15:14, 387 B 2 minutes ago
External Libraries
Scratches and Consoles
1
2
Sync Now
Ignore these c

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

```
pluginManagement { PluginManagementSpec it ->
    repositories { RepositoryHandler it ->
        gradlePluginPortal()
        google()
        mavenCentral()
    }
}
dependencyResolutionManagement { DependencyResolutionManagement it ->
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories { RepositoryHandler it ->
        google()
        mavenCentral()
        flatDir {
            dirs 'libs'
        }
    }
}
rootProject.name = "TencentLiveSDKDemo"
include ':app'
```

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


TencentLiveSDKDemo – build.gradle (:app)
TencentLiveSDKDemo 〉 app 〉 build.gradle
app
No Devices
Project
activity_main.xml
MainActivity.java
build.gradle (TencentLiveSDKDemo)
settings.gradle (TencentLiveSDKDemo)
gradle.properties
build.gradle (:app)
Sync Now
Ignore these c
1
2
3
4
dependencies{}
系统设置

TencentLiveSDKDemo ~/AndroidStudioProjects/TencentLiv
.gradle
.idea
app
libs
src
.gitignore 2023/6/20, 14:54, 6 B
build.gradle 2023/6/20, 15:18, 1.22 kB Moments ago
proguard-rules.pro 2023/6/20, 14:54, 750 B
gradle
.gitignore 2023/6/20, 14:54, 225 B 18 minutes ago
build.gradle 2023/6/20, 14:54, 229 B 4 minutes ago
gradle.properties 2023/6/20, 14:54, 1.27 kB 4 minutes ago
gradlew 2023/6/20, 14:54, 5.77 kB
gradlew.bat 2023/6/20, 14:54, 2.76 kB
local.properties 2023/6/20, 14:54, 434 B
settings.gradle 2023/6/20, 15:14, 387 B A minute ago
External Libraries
Scratches and Consoles

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

```
android {
    namespace 'com.example.tencentlivesdkdemo'
    compileSdk 32

    defaultConfig {
        applicationId "com.example.tencentlivesdkdemo"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        ndk {
            abiFilters "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_Live_11.2.0.13154', ext:'aar')
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Gradle sync finished in 5 s 666 ms (13 minutes ago)
43:2 LF UTF-8 4 spaces

```
implementation(name:'LiteAVSDK_Live_11.2.0.13154', 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.

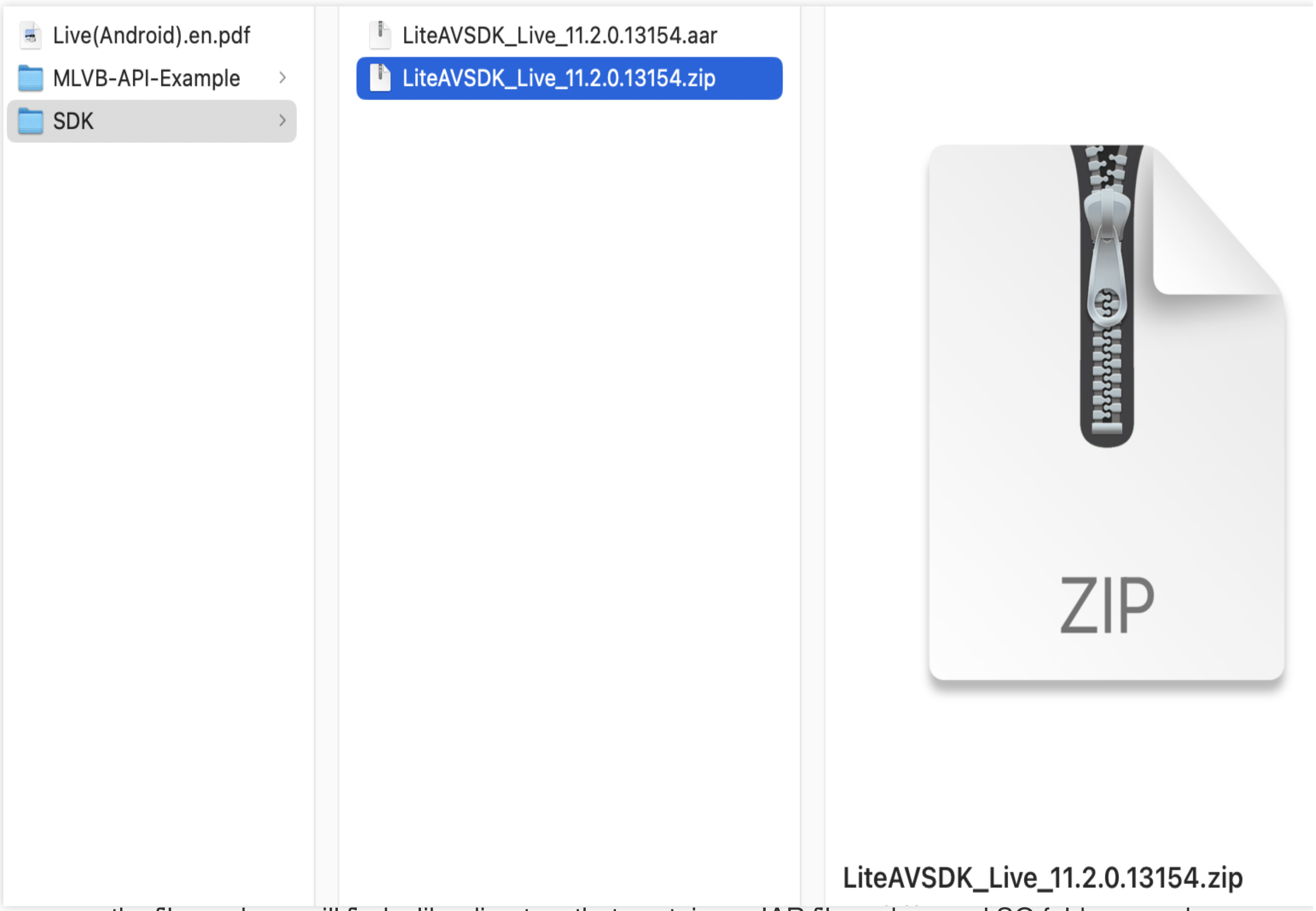
```
defaultConfig {
     ndk {
         abiFilters "armeabi-v7a", "arm64-v8a"
     }
}
```

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

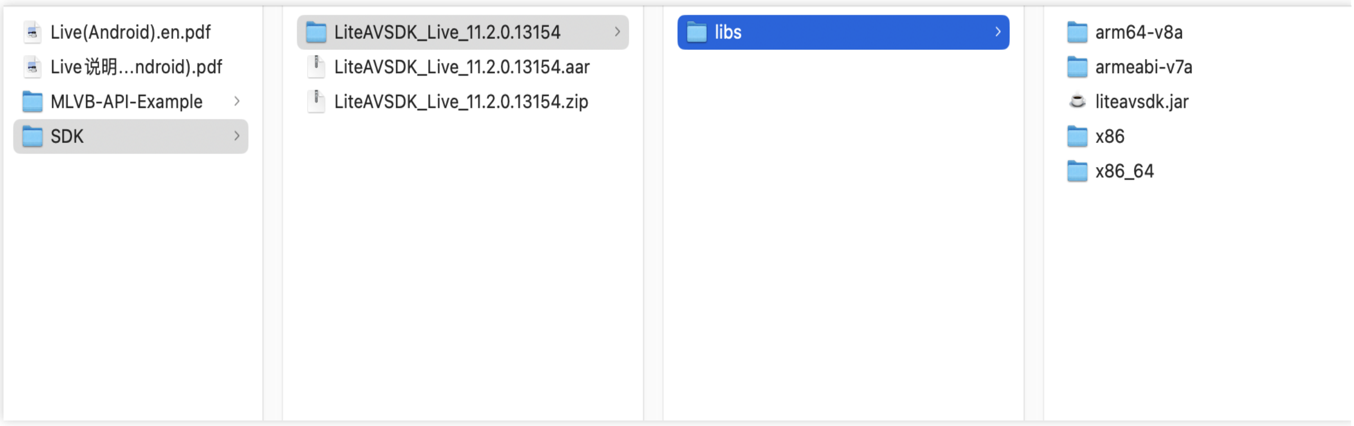
Integrating the SDK (JAR)

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

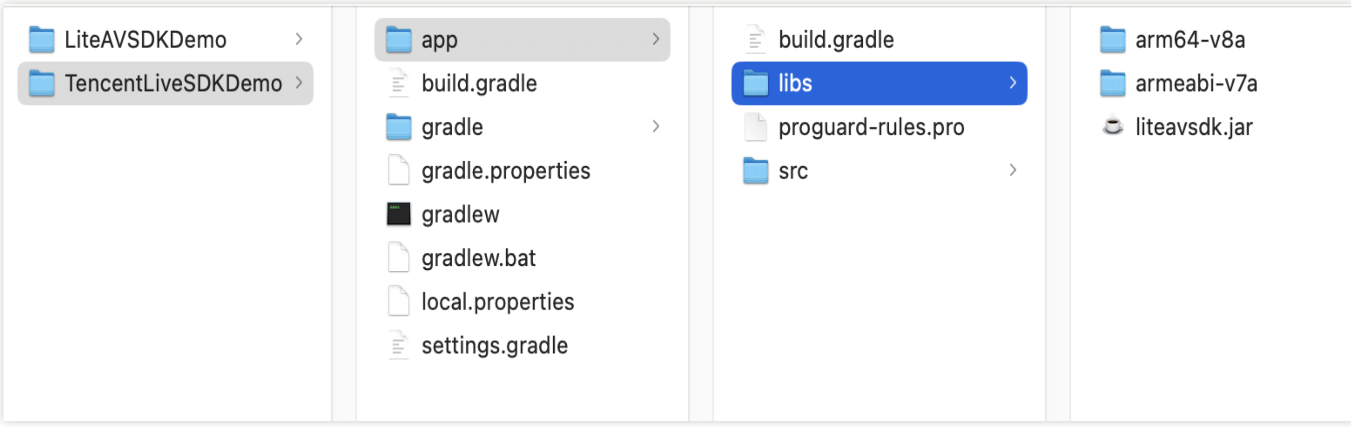
1. Download LiveAVSDK and decompress the file. In the SDK directory, find LiteAVSDK_Live_xxx.zip (xxx indicates the version number of LiteAVSDK).


Live(Android).en.pdf
MLVB-API-Example
SDK
LiteAVSDK_Live_11.2.0.13154.aar
LiteAVSDK_Live_11.2.0.13154.zip
ZIP
LiteAVSDK_Live_11.2.0.13154.zip

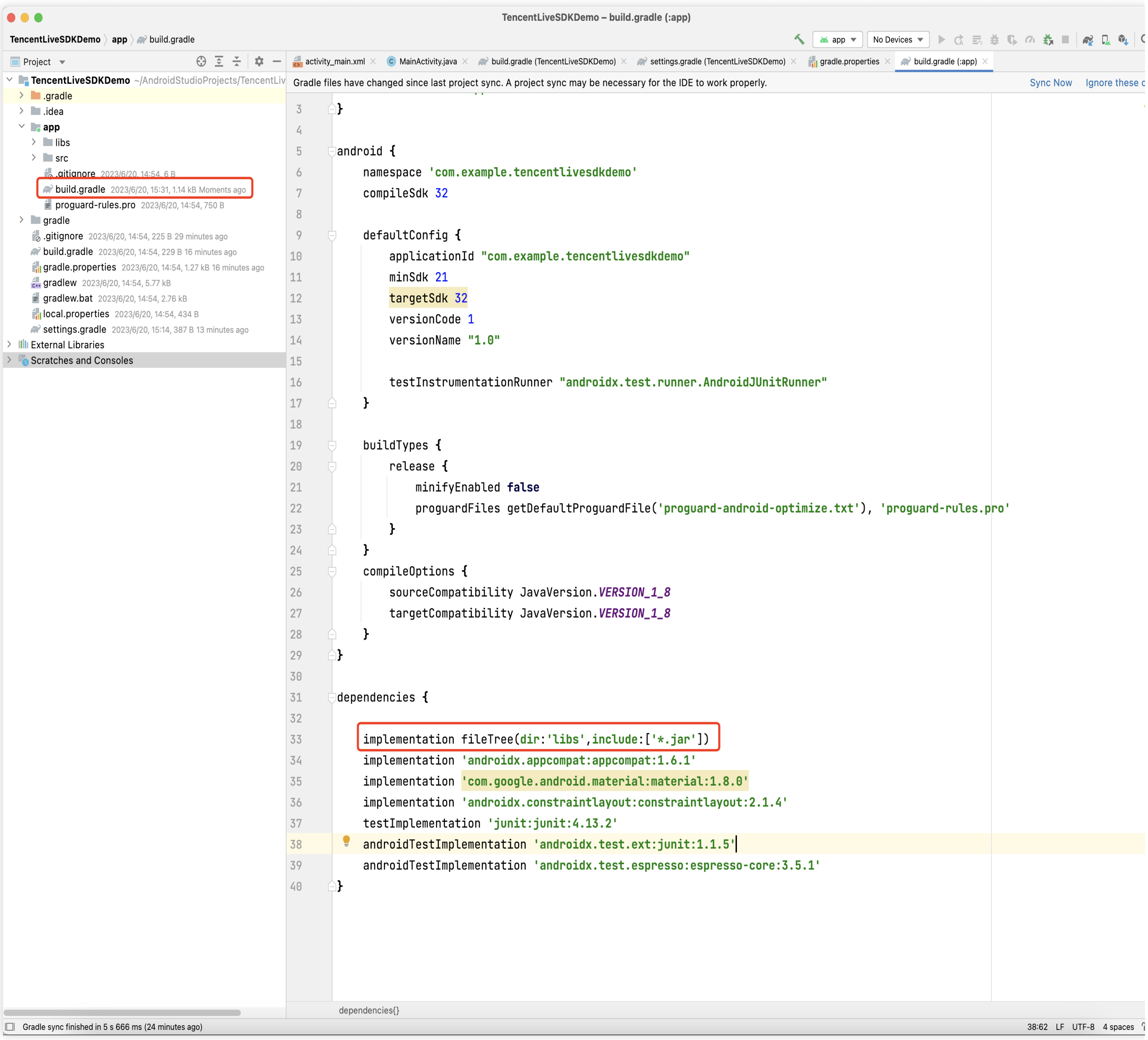
2. Decompress the file, and you will find a libs directory that contains a JAR file and several SO folders, as shown below:



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



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

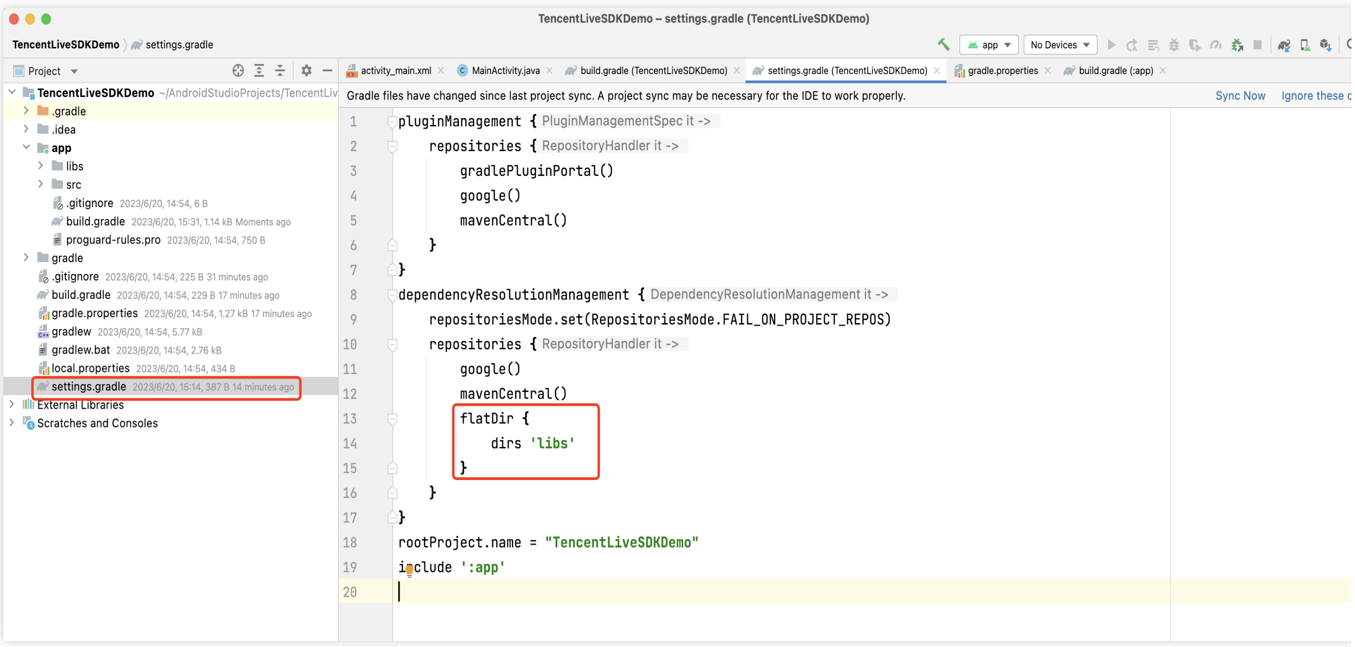

TencentLiveSDKDemo – build.gradle (:app)
TencentLiveSDKDemo › app › build.gradle
Project
activity_main.xml
MainActivity.java
build.gradle (TencentLiveSDKDemo)
settings.gradle (TencentLiveSDKDemo)
gradle.properties
build.gradle (:app)
TencentLiveSDKDemo ~/AndroidStudioProjects/TencentLiv
.gradle
.idea
app
libs
src
.gitignore 2023/6/20, 14:54, 6 B
build.gradle 2023/6/20, 15:31, 1.14 kB Moments ago
proguard-rules.pro 2023/6/20, 14:54, 750 B
gradle
.gitignore 2023/6/20, 14:54, 225 B 29 minutes ago
build.gradle 2023/6/20, 14:54, 229 B 16 minutes ago
gradle.properties 2023/6/20, 14:54, 1.27 kB 16 minutes ago
gradlew 2023/6/20, 14:54, 5.77 kB
gradlew.bat 2023/6/20, 14:54, 2.76 kB
local.properties 2023/6/20, 14:54, 434 B
settings.gradle 2023/6/20, 15:14, 387 B 13 minutes ago
External Libraries
Scratches and Consoles
Sync Now
Ignore these c
dependencies{}
Gradle sync finished in 5 s 666 ms (24 minutes ago)
38:62 LF UTF-8 4 spaces

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

```
}

android {
    namespace 'com.example.tencentlivesdkdemo'
    compileSdk 32

    defaultConfig {
        applicationId "com.example.tencentlivesdkdemo"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    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 fileTree(dir:'libs',include:['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

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

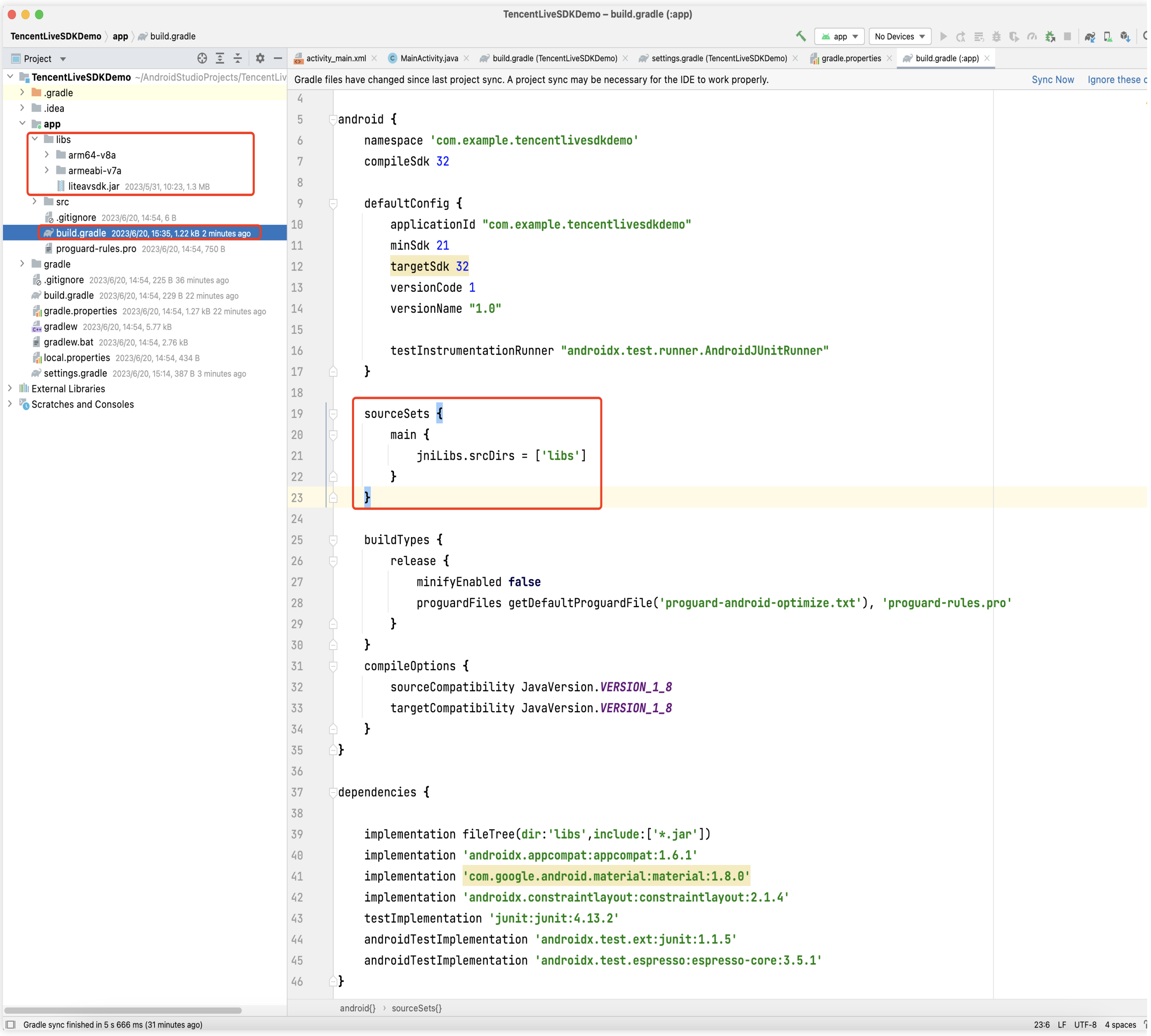
5. Add **flatDir** to `build.gradle` under the project’s root directory to specify the local path for the repository.


TencentLiveSDKDemo – settings.gradle (TencentLiveSDKDemo)
TencentLiveSDKDemo
settings.gradle
app
No Devices
Project
activity_main.xml
MainActivity.java
build.gradle (TencentLiveSDKDemo)
settings.gradle (TencentLiveSDKDemo)
gradle.properties
build.gradle (:app)
TencentLiveSDKDemo ~/AndroidStudioProjects/TencentLiv
.gradle
.idea
app
libs
src
.gitignore 2023/6/20, 14:54, 6 B
build.gradle 2023/6/20, 15:31, 1.14 kB Moments ago
proguard-rules.pro 2023/6/20, 14:54, 750 B
gradle
.gitignore 2023/6/20, 14:54, 225 B 31 minutes ago
build.gradle 2023/6/20, 14:54, 229 B 17 minutes ago
gradle.properties 2023/6/20, 14:54, 1.27 kB 17 minutes ago
gradlew 2023/6/20, 14:54, 5.77 kB
gradlew.bat 2023/6/20, 14:54, 2.76 kB
local.properties 2023/6/20, 14:54, 434 B
settings.gradle 2023/6/20, 15:14, 387 B 14 minutes ago
External Libraries
Scratches and Consoles
Sync Now
Ignore these c

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

```
pluginManagement { PluginManagementSpec it ->
    repositories { RepositoryHandler it ->
        gradlePluginPortal()
        google()
        mavenCentral()
    }
}
dependencyResolutionManagement { DependencyResolutionManagement it ->
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories { RepositoryHandler it ->
        google()
        mavenCentral()
        flatDir {
            dirs 'libs'
        }
    }
}
rootProject.name = "TencentLiveSDKDemo"
include ':app'

```

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


TencentLiveSDKDemo – build.gradle (:app)
TencentLiveSDKDemo 〉 app 〉 build.gradle
app
No Devices
Project
activity_main.xml
MainActivity.java
build.gradle (TencentLiveSDKDemo)
settings.gradle (TencentLiveSDKDemo)
gradle.properties
build.gradle (:app)
Sync Now
Ignore these c

```
TencentLiveSDKDemo ~/AndroidStudioProjects/TencentLiv
  .gradle
  .idea
  app
    libs
      arm64-v8a
      armeabi-v7a
      liteavsdk.jar 2023/5/31, 10:23, 1.3 MB
    src
    .gitignore 2023/6/20, 14:54, 6 B
    build.gradle 2023/6/20, 15:35, 1.22 kB 2 minutes ago
    proguard-rules.pro 2023/6/20, 14:54, 750 B
  gradle
  .gitignore 2023/6/20, 14:54, 225 B 36 minutes ago
  build.gradle 2023/6/20, 14:54, 229 B 22 minutes ago
  gradle.properties 2023/6/20, 14:54, 1.27 kB 22 minutes ago
  gradlew 2023/6/20, 14:54, 5.77 kB
  gradlew.bat 2023/6/20, 14:54, 2.76 kB
  local.properties 2023/6/20, 14:54, 434 B
  settings.gradle 2023/6/20, 15:14, 387 B 3 minutes ago
External Libraries
Scratches and Consoles
```

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

```
android {
    namespace 'com.example.tencentlivesdkdemo'
    compileSdk 32

    defaultConfig {
        applicationId "com.example.tencentlivesdkdemo"
        minSdk 21
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

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

    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 fileTree(dir:'libs',include:['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

android{} › sourceSets{}

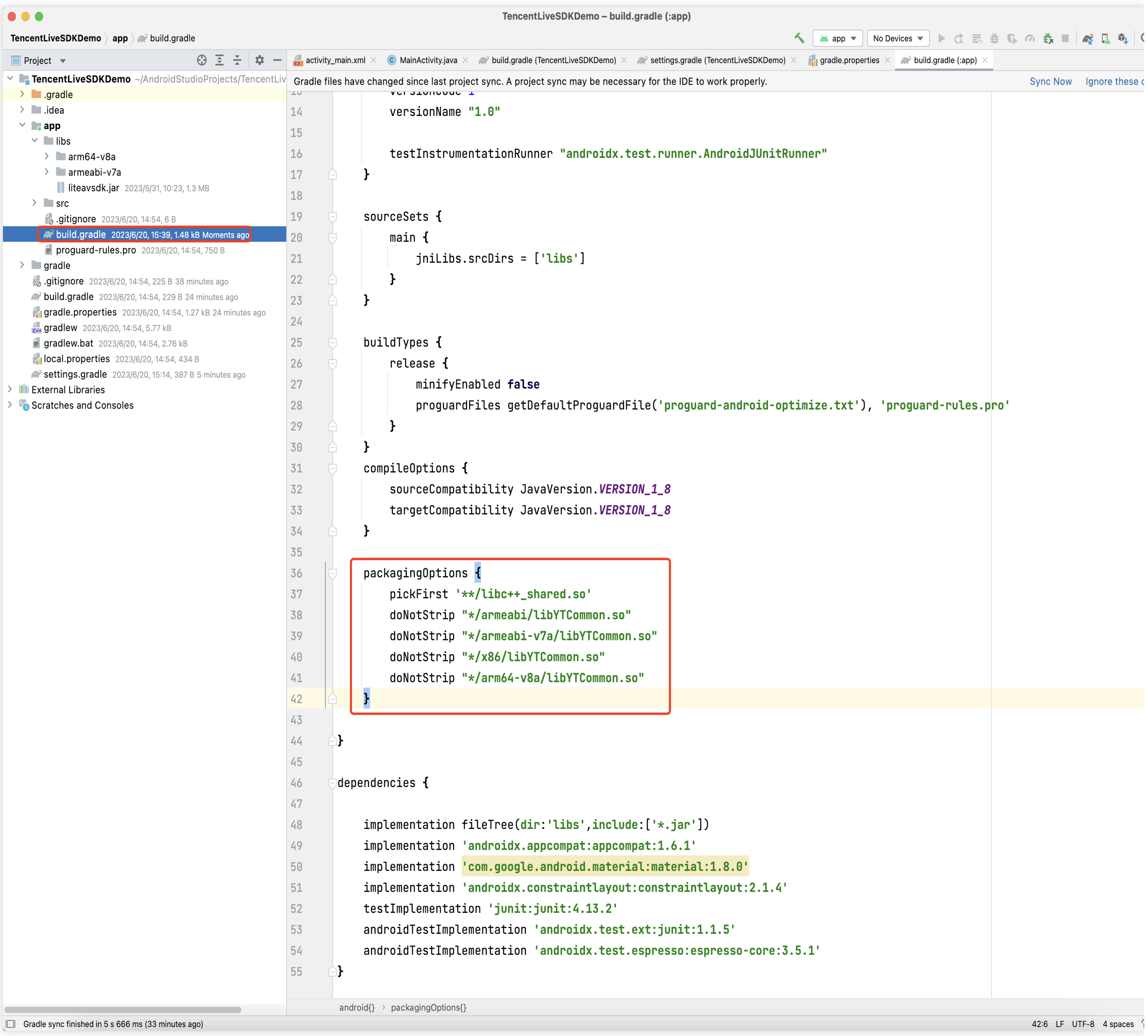
Gradle sync finished in 5 s 666 ms (31 minutes ago)   23:6 LF UTF-8 4 spaces

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

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

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

Setting Packaging Parameters


TencentLiveSDKDemo – build.gradle (:app)
TencentLiveSDKDemo 〉 app 〉 build.gradle
app
No Devices
Project
activity_main.xml
MainActivity.java
build.gradle (TencentLiveSDKDemo)
settings.gradle (TencentLiveSDKDemo)
gradle.properties
build.gradle (:app)
TencentLiveSDKDemo ~/AndroidStudioProjects/TencentLiv
.gradle
.idea
app
libs
arm64-v8a
armeabi-v7a
liteavsdk.jar 2023/5/31, 10:23, 1.3 MB
src
.gitignore 2023/6/20, 14:54, 6 B
build.gradle 2023/6/20, 15:39, 1.48 kB Moments ago
proguard-rules.pro 2023/6/20, 14:54, 750 B
gradle
.gitignore 2023/6/20, 14:54, 225 B 38 minutes ago
build.gradle 2023/6/20, 14:54, 229 B 24 minutes ago
gradle.properties 2023/6/20, 14:54, 1.27 kB 24 minutes ago
gradlew 2023/6/20, 14:54, 5.77 kB
gradlew.bat 2023/6/20, 14:54, 2.76 kB
local.properties 2023/6/20, 14:54, 434 B
settings.gradle 2023/6/20, 15:14, 387 B 5 minutes ago
External Libraries
Scratches and Consoles
Sync Now
Ignore these c
android{} 〉 packagingOptions{}
42:6 LF UTF-8 4 spaces

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

```
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

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

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

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

}

dependencies {

    implementation fileTree(dir:'libs',include:['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.8.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Gradle sync finished in 5 s 666 ms (33 minutes ago)

```
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.autofocus" />
```

Configuring License

Click Get License to obtain a trial license. You will get two strings: a license URL and a decryption key.
Before you use the features of MLVB Enterprise Edition 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 = ""; // your licence url
        String licenceKey = ""; // your licence key
        V2TXLivePremier.setEnvironment("GDPR"); // set environment
        V2TXLivePremier.setLicence(this, licenceURL, licenceKey);
        V2TXLivePremier.setObserver(new V2TXLivePremierObserver() {
            @Override
            public void onLicenceLoaded(int result, String reason) {
                Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reas
            }
        });
    }
}
```

Configuring Obfuscation Rules

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

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

FAQs

1. How to solve the crash problem when using the LiteAVSDK screen recording/screen sharing function on the Android side?

Please check the targetSdkVersion setting in the project first. If it is set to 29, then running Android 10 devices using screen recording and sharing will trigger a flashback problem. The reason is that the Android privacy policy has changed. The solution needs to start the foreground service and specify the type as mediaProjection. There is no need to call startScreenCapture in the Service.

2. How to solve the problem of symbol conflict between multiple SDKs of LiteAVSDK series such as live SDK/real-time audio/video/player integrated in the project?

If you integrate 2 or more LiteAVSDK products (live broadcast, player, TRTC, short video), there will be a library conflict problem when compiling, because some SDK underlying libraries have the same symbol files, it is recommended to integrate only one full-featured version of the SDK. Live broadcast, player, TRTC, and short video are all included in one SDK.For details, please refer to SDK Download.

Flutter

Last updated：2024-01-13 15:49:41

This document describes how to quickly integrate live_flutter_plugin (Tencent Cloud RT-Cube MLVB Flutter plugin) into your project. For the demo project, visit GitHub.

Environment Requirements

Flutter 2.0 or later

Developing for Android:

Android Studio 3.5 or later

Devices with Android 4.1 or later

Developing for iOS and macOS:

Xcode 11.0 or later

OS X 10.11 or later

A valid developer signature for your project

Quickly Integrating the SDK

The SDK for Flutter has been released to the Pub repository. You can configure `pubspec.yaml` to download the update automatically.

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

```
dependencies:
  live_flutter_plugin: latest version number
```

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

iOS

Android

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>You can make audio calls only if you grant the app mic permission.</string>
```

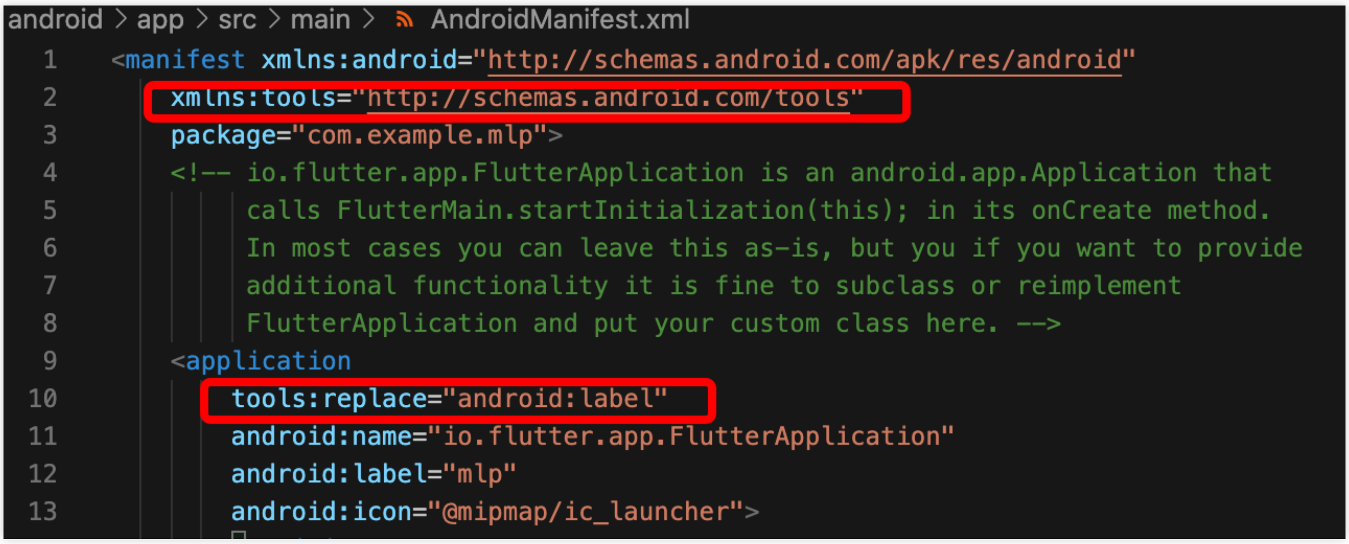
2. Get camera and mic permissions to enable the audio and video call features.

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

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

Getting Started

1. Click Apply for License to get a trial license. You will get two strings: a license URL and a decryption key.
2. Before your application calls features of `live_flutter_plugin`, complete the following configuration:

```
import 'package:live_flutter_plugin/v2_tx_live_premier.dart';

 /// Tencent Cloud license management page (https://console.tencentcloud.com/live/l
setupLicense() {
  // The license URL of the current application
  var LICENSEURL = "";
  // The license key of the current application
  var LICENSEURLKEY = "";
  V2TXLivePremier.setLicence(LICENSEURL, LICENSEURLKEY);
}
```

FAQs

For more FAQs, see Flutter.

How do I get a valid stream push URL?

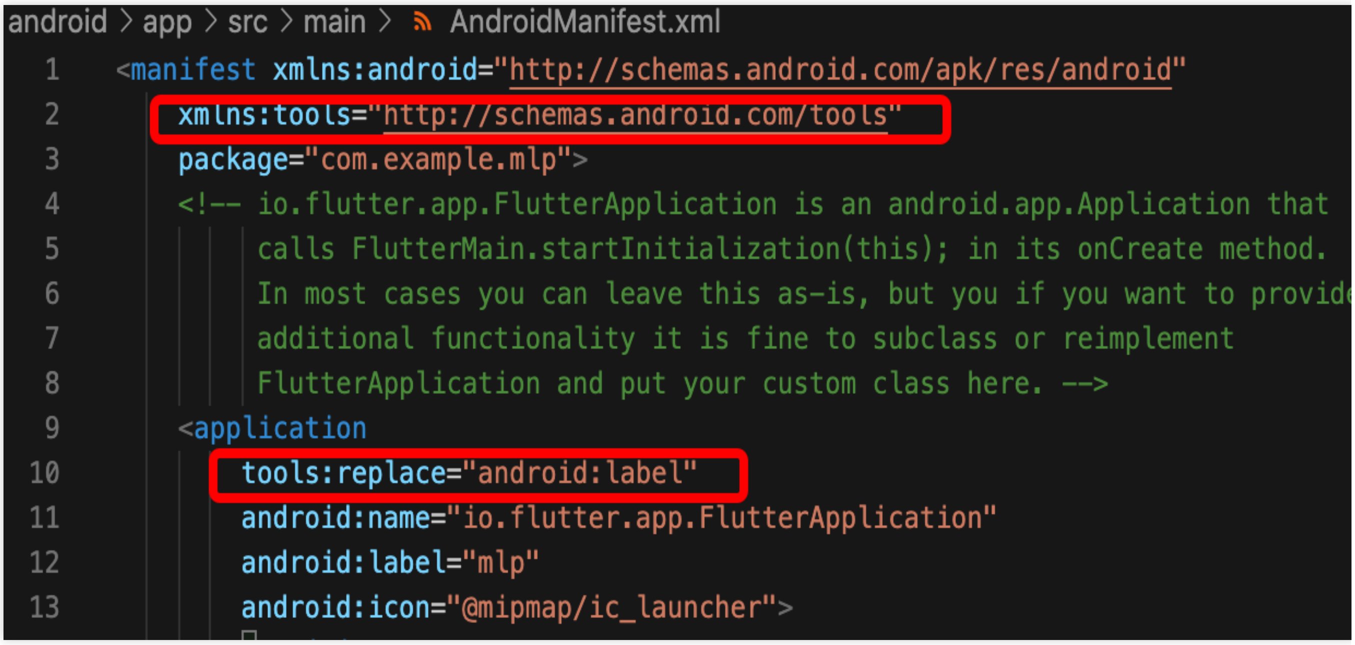
Activate CSS. In the **CSS console**, go to **Auxiliary Tools** > **Address Generator** to generate a stream push URL. For more information, see Publishing/Playback URL.

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

Check whether `io.flutter.embedded_views_preview` in `info.plist` is `YES` .

What should I do if the "Manifest merge failed" error occurs in Android Studio?

1. Open `/example/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` .



android > app > src > main > AndroidManifest.xml

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

Publishing

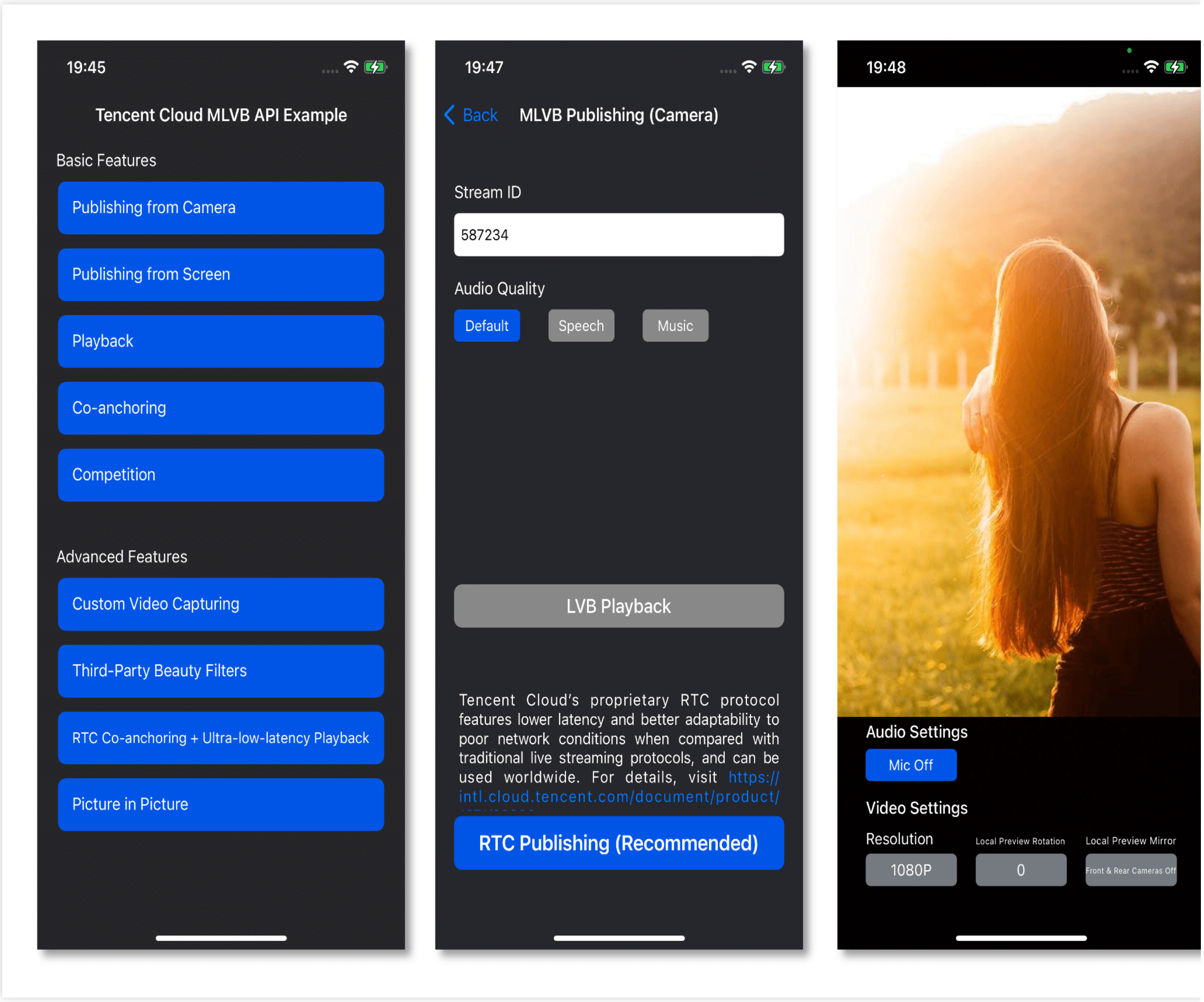
iOS

Publishing from Camera

Last updated：2024-01-13 15:49:41

Overview

Publishing from camera refers to the process of collecting video and audio data from the mobile phone's camera and mic, encoding the data, and publishing it to cloud-based live streaming platforms. Tencent Cloud's LiteAVSDK provides the camera publishing capability via `V2TXLivePusher` , the following is the relevant operation interface of the demo camera in the simple version of LiteAVSDK:



Tencent Cloud's proprietary RTC protocol features lower latency and better adaptability to poor network conditions when compared with traditional live streaming protocols, and can be used worldwide. For details, visit https://intl.cloud.tencent.com/document/product/

Notes

About running projects on x86 emulators: The SDK uses a lot of audio and video APIs of the iOS system, most of which cannot be used on the x86 emulator built into macOS. Therefore, we recommend that you test your project on a real device.

Sample Code

Platform	GitHub Address	Key Class
iOS	Github	CameraPushViewController.m

Android	Github	CameraPushMainActivity.java
Flutter	Github	live_camera_push.dart

Integration

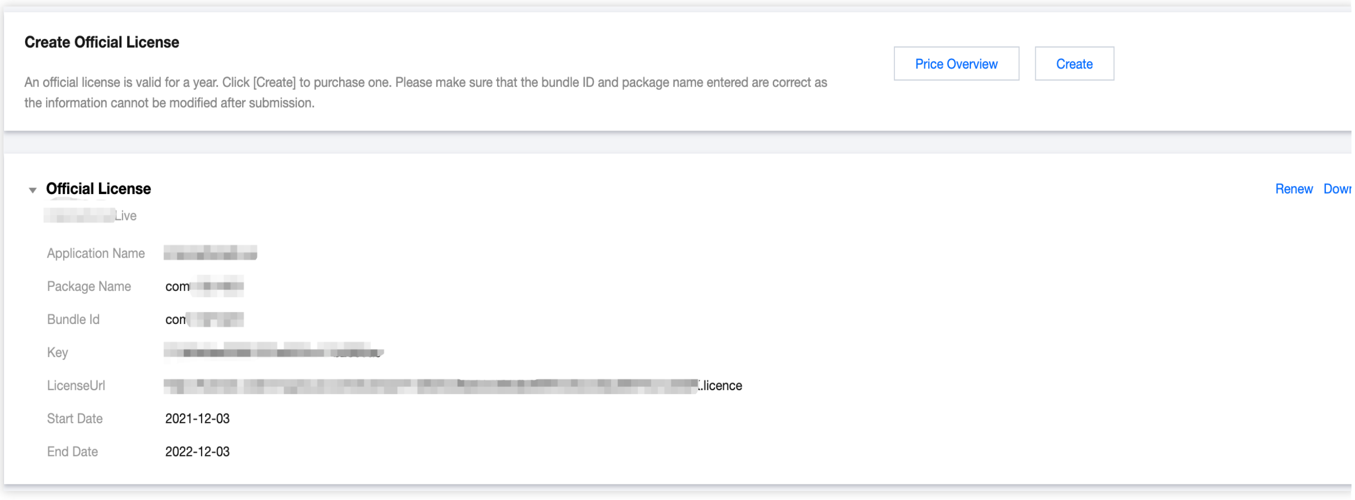
1. Download the SDK

Download the SDK and follow the instructions in SDK Integration to integrate the SDK into your application.

2. Configure License Authorization for SDK

1. Obtain license authorization：

If you have obtained the relevant license authorization， Need to Get License URL and License Key in Cloud Live Console



Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

[redacted]Live

Application Name	[redacted]
Package Name	com[redacted]
Bundle Id	con[redacted]
Key	[redacted]
LicenseUrl	[redacted].licence
Start Date	2021-12-03
End Date	2022-12-03

If you have not yet obtained the license authorization， Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the `- [AppDelegate application:didFinishLaunchingWithOptions:]`), set the following settings:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<your licenseUrl>";
    NSString * const licenceKey = @"<your key>";

    // V2TXLivePremier is located in "V2TXLivePremier.h"
    [V2TXLivePremier setEnvironment:@"GDPR"]; // set environment
    [V2TXLivePremier setLicence:licenceURL key:licenceKey];
    [V2TXLivePremier setObserver:self];
    NSLog(@"SDK Version = %@", [V2TXLivePremier getSDKVersionStr]);
    return YES;
}

#pragma mark - V2TXLivePremierObserver
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
}
@end
```

Note:

The BundleId configured in the license must be the same as the application itself, otherwise the streaming will fail.

3. Initialize the `V2TXLivePusher` component

Create a `V2TXLivePusher` object and specify `V2TXLiveMode` .

```
// Specify the corresponding live broadcast protocol as RTMP, which does not suppor
V2TXLivePusher *pusher = [[V2TXLivePusher alloc] initWithLiveMode:V2TXLiveMode_RTMP
```

4. Enable camera preview

Call `setRenderView` in V2TXLivePusher to configure a view object for displaying video images, and then call `startCamera` to enable camera preview for your mobile phone.

```
 // Create a view object and insert it into the UI
 UIView *_localView = [[UIView alloc] initWithFrame:self.view.bounds];
 [self.view insertSubview:_localView atIndex:0];
 _localView.center = self.view.center;

 // Enable preview for the local camera
 [_pusher setRenderView:_localView];
 [_pusher startCamera:YES];
 [_pusher startMicrophone];
```

Note:

To add animated effects to the view, modify its `transform` attribute rather than `frame` attribute.

```
[UIView animateWithDuration:0.5 animations:^{
      _localView.transform = CGAffineTransformMakeScale(0.3, 0.3); // Shrink by 1/3
}];
```

5. Start and stop publishing

After calling `startCamera` to enable camera preview, you can call the startPush API in `V2TXLivePusher` to start publishing.

Note:

If the `RTMP` protocol is selected in Step 3 to push the stream, please refer to the generation of the push stream URL RTMP URL。

```
//This URL does not support co-anchoring. The stream is published to a live streami
NSString* url = @"rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx";
[_pusher startPush:url];
```

Call stopPush in `V2TXLivePusher` to stop publishing streams.

```
//Stop publishing
[_pusher stopPush];
```

Why is `V2TXLIVE_ERROR_INVALID_LICENSE` returned?If the `startPush` API returns `V2TXLIVE_ERROR_INVALID_LICENSE` , it means your license verification failed. Please check your configuration against Step 2. Configure License Authorization for SDK.

6. Publish audio-only streams

If your live streaming scenarios involve audio only, you can skip Step 4 or do not call `startCamera` before `startPush` .

```
[_pusher startMicrophone];
//This URL does not support co-anchoring. The stream is published to a live streami
NSString* url = @"rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx";
[_pusher startPush:url];
```

Note:

If you publish audio-only streams but no streams can be pulled from an RTMP, FLV, or HLS playback URL, there is a problem with your line configuration, please submit a ticket for help.

7. Set video quality

Call setVideoQuality in `V2TXLivePusher` to set the quality of videos watched by audience. The encoding parameters set determine the quality of videos presented to audience. The local video watched by the host is the original HD version that has not been encoded or compressed, and is therefore not affected by the settings. For details, please see Setting Video Quality.

8. Set the beauty filter style and skin brightening and rosy skin effects

Call getBeautyManager in `V2TXLivePusher` to get a `TXBeautyManager` instance to set beauty filters.

Beauty filter style

The SDK has three built-in beauty filter algorithms, each corresponding to a beauty filter style. Choose one that best fits your product positioning. For details, please see the TXBeautyManager.h file.

Beauty Filter Style	Description
TXBeautyStyleSmooth	The smooth style, which features more obvious skin smoothing effects and is suitable for live showrooms
TXBeautyStyleNature	The natural style, which retains more facial details and is more natural
TXBeautyStylePitu	The Pitu style, which uses the beauty filter algorithm developed by YouTu Lab. Its effect is between the smooth style and the natural style, that is, it retains more skin details than the smooth style and delivers more obvious skin smoothing effects than the natural style.

You can call the setBeautyStyle API in `TXBeautyManager` to set the beauty filter style.

Item	Configuration	Description
Beauty filter strength	Via the setBeautyLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the more obvious the effect.
Skin brightening filter strength	Via the setWhitenessLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the more obvious the effect.
Rosy skin filter strength	Via the setRuddyLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the more obvious the effect.

9. Set color filters

Call getBeautyManager in `V2TXLivePusher` to get a `TXBeautyManager` instance to set color filters.
Call the `setFilter` API in `TXBeautyManager` to set color filters. Color filters are a technology that adjusts the color tone of sections of an image. For example, it may lighten the yellow sections of an image to achieve the effect of skin brightening, or add warm tones to a video to give it a refreshing and soft boost.
Call the `setFilterStrength` API in `TXBeautyManager` to set the strength of a color filter. The higher the strength, the more obvious the effect.
Based on our experience of operating Mobile QQ and Now Live, it's not enough to use only the `setBeautyStyle` API in `TXBeautyManager` to set the beauty filter style. The `setBeautyStyle` API must be used together with `setFilter` to produce richer effects. Given this, our designers have developed 17 built-in color filters for you to choose from.

```
NSString * path = [[NSBundle mainBundle] pathForResource:@"FilterResource" ofType:@
path = [path stringByAppendingPathComponent:lookupFileName];
```

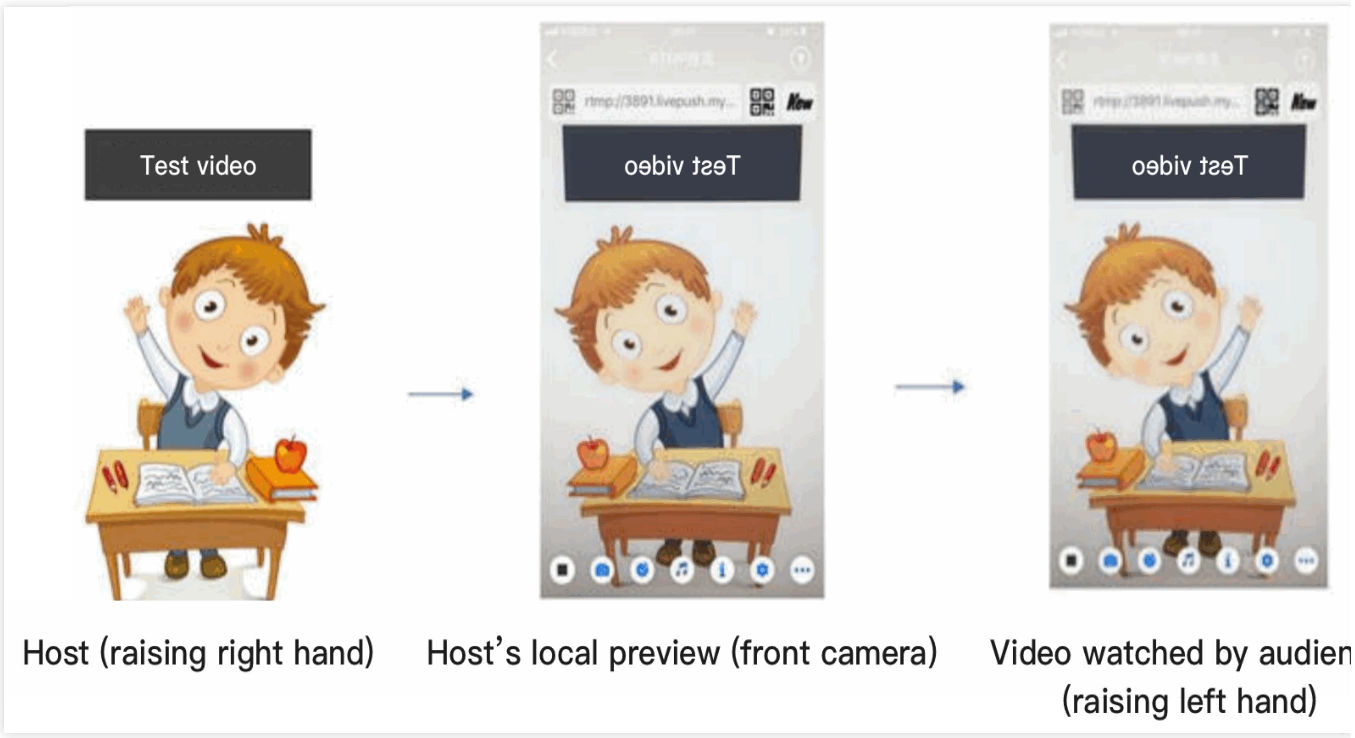
```
UIImage *image = [UIImage imageWithContentsOfFile:path];
[[_pusher getBeautyManager] setFilter:image];
[[_pusher getBeautyManager] setFilterStrength:0.5f];
```

10. Manage devices

`V2TXLivePusher` provides a series of APIs for the control of devices. You can call `getDeviceManager` to get a `TXDeviceManager` instance for device management. For detailed instructions, please see TXDeviceManager API.

11. Set the video mirroring effect for audience

Call setEncoderMirror in `V2TXLivePusher` to set the camera mirror mode, which affects the way video images are presented to audience. By default, the local image seen by the host is flipped when the front camera is used.



12. Publish streams in landscape mode

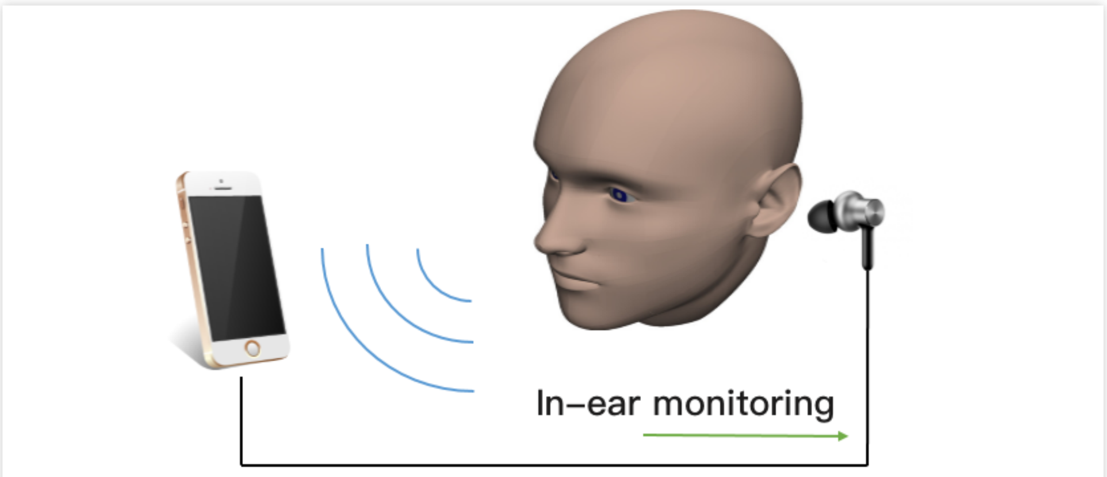
In most cases, hosts stream while holding their phones vertically, and audience watch videos in portrait resolutions (e.g., 540 × 960). However, there are also cases where hosts hold phones horizontally, and ideally, audience should watch videos in landscape resolutions (960 × 540).

By default, `V2TXLivePusher` outputs videos in portrait resolutions. You can publish landscape-mode videos to audience by modifying a parameter of the setVideoQuality API.

```
[_pusher setVideoQuality:videoQuality
          resolutionMode:isLandscape ? V2TXLiveVideoResolutionModeLandscape : V2
```

13. Set audio effects

Call getAudioEffectManager in `V2TXLivePusher` to get a `TXAudioEffectManager` instance, which can be used to mix background music and set in-ear monitoring, reverb, and other audio effects. Background music mixing means mixing into the published stream the music played by the host’s phone so that audience can also hear the music.
Call the `enableVoiceEarMonitor` API in `TXAudioEffectManager` to enable in-ear monitoring, which allows hosts to hear their vocals in earphones when they sing.
Call the `setVoiceReverbType` API in `TXAudioEffectManager` to add reverb effects such as karaoke, hall, husky, and metal. The effects are applied to the videos watched by audience.
Call the `setVoiceChangerType` API in `TXAudioEffectManager` to add voice changing effects such as little girl and middle-aged man to enrich host-audience interaction. The effects are applied to the videos watched by audience.



In–ear Monitoring Illustration

Note:
For detailed instructions, please see TXAudioEffectManager API.

14. Set watermarks

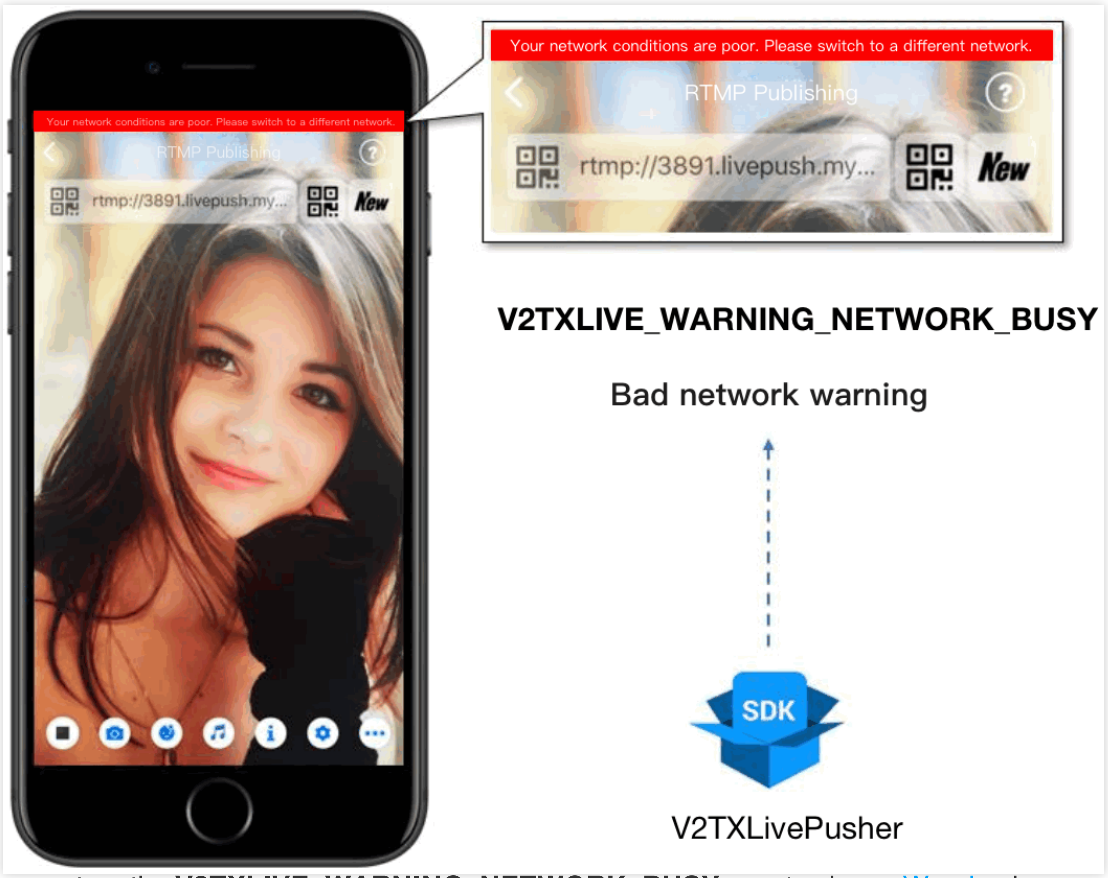
Call setWatermark in `V2TXLivePusher` to add a watermark to videos output by the SDK. The position of the watermark is determined by the `(x, y, scale)` parameter passed in.
The watermark image must be in PNG rather than JPG format. The former carries opacity information, which allows the SDK to better address the image aliasing issue (changing the extension of a JPG image to PNG won’t work).
The `(x, y, scale)` parameter specifies the normalized coordinates of the watermark relative to the resolution of the published video. For example, if the resolution of the published video is 540 x 960, and `(x, y, scale)` is set to `(0.1, 0.1, 0.1)`, the actual pixel coordinates of the watermark will be (540 x 0.1, 960 x 0.1). The width of the watermark will be the video width x 0.1, and the height will be scaled automatically.

```
// Set a video watermark
```

```
[_pusher setWatermark:[UIImage imageNamed:@"watermark"] x:0.03 y:0.015 scale:1];
```

15. Inform hosts of poor network conditions

Connecting phones to Wi-Fi does not necessarily guarantee network conditions. In case of poor Wi-Fi signal or limited bandwidth, the network speed of a Wi-Fi connected phone may be slower than that of a phone using 4G. Hosts should be informed when their network conditions are bad and be prompted to switch to a different network.


Your network conditions are poor. Please switch to a different network.
RTMP Publishing
rtmp://3891.livepush.my...
New
V2TXLIVE_WARNING_NETWORK_BUSY
Bad network warning
SDK
V2TXLivePusher

You can capture the **V2TXLIVE_WARNING_NETWORK_BUSY** event using onWarning in `V2TXLivePusherObserver` . The event indicates poor network conditions for hosts, which result in stuttering for audience. When this event occurs, you can send a UI message about poor network conditions to hosts, as shown above.

```
- (void)onWarning:(V2TXLiveCode)code
          message:(NSString *)msg
        extraInfo:(NSDictionary *)extraInfo {
    dispatch_async(dispatch_get_main_queue(), ^{
        if (code == V2TXLIVE_WARNING_NETWORK_BUSY) {
            [_notification displayNotificationWithMessage:
                @"Your network conditions are poor. Please switch to a
different network." forDuration:5];
        }
    });
}
```

16. Send SEI messages

Call the sendSeiMessage API in `V2TXLivePusher` to send SEI messages. SEI refers to the supplementary enhancement information of encoded video. It is not used most of the time, but you can insert custom information into SEI messages. The information will be forwarded to audience by live streaming CDNs. The applications for SEI messages include:
Live quiz: The publisher can use SEI messages to send questions to the audience. SEI can ensure synchronization among audio, video, and the questions.
Live showroom: The publisher can use SEI messages to display lyrics to the audience in real time. The effects are not affected by reduction in video encoding quality.
Online education: The publisher can use SEI messages to display pointers and sketches on slides to the audience in real time.
Custom data is inserted directly into video data and therefore cannot be too large in size (preferably several bytes). It's common to insert information such as custom timestamps.

```
int payloadType = 5;
NSString* msg = @"test";
[_pusher sendSeiMessage:payloadType data:[msg dataUsingEncoding:NSUTF8StringEncodin
```

Common open-source players or web players are incapable of parsing SEI messages. You must use `V2TXLivePlayer` , the built-in player of LiteAVSDK.

1. Configuration:

```
int payloadType = 5;
[_player enableReceiveSeiMessage:YES payloadType:payloadType];
```

2. If the video streams played by `V2TXLivePlayer` contain SEI messages, you will receive the messages via the `onReceiveSeiMessage` callback in `V2TXLivePlayerObserver` .

Event Handling

Listening for events

The SDK listens for publishing events and errors via the V2TXLivePusherObserver delegate. See V2TXLiveCode for a detailed list of events and error codes.

Errors

An error indicates that the SDK encountered a serious problem that made it impossible for stream publishing to continue.

Event ID	Code	Description
V2TXLIVE_ERROR_FAILED	-1	A common error not yet classified
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	An invalid parameter was passed in during API calling.
V2TXLIVE_ERROR_REFUSED	-3	The API call was rejected.
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	The API cannot be called.
V2TXLIVE_ERROR_INVALID_LICENSE	-5	Failed to call the API due to invalid license.
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	The server request timed out.
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	The server could not handle your request.

Warnings

A warning indicates that the SDK encountered a problem whose severity level is warning. Warning events trigger tentative protection or recovery logic and can often be resolved.

Event ID	Code	Description
V2TXLIVE_WARNING_NETWORK_BUSY	1101	Bad network connection: data upload blocked due to limited upstream bandwidth.
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	Latency during video playback.
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	Failed to turn the camera on.
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	The camera is occupied. Try a different camera.
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	No access to the camera. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	Failed to turn the mic on.
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	The mic is occupied. This occurs when, for example, the user is having a call on the mobile device.

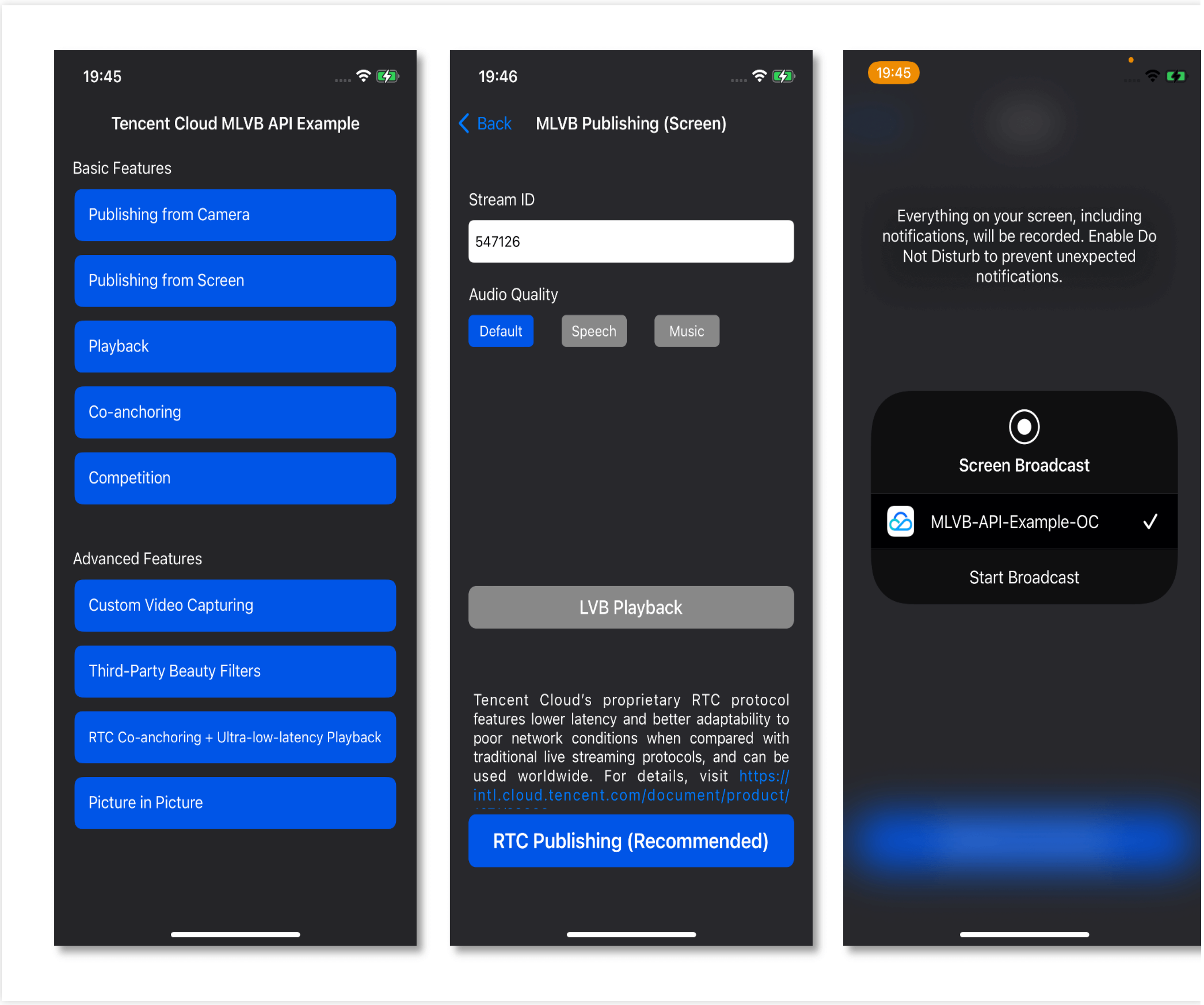
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	No access to the mic. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	The system does not support screen sharing.
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	Failed to start screen recording. If this occurs on a mobile device, it may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	Screen recording was stopped by the system.

Publishing from Screen

Last updated：2024-01-13 15:49:41

Overview

Screen recording allows a host to live stream the image on their phone screen. It can be combined with local camera preview and is used in scenarios such as game streaming and mobile app demos. The Tencent Cloud LiteAV SDK offers screen recording capabilities via `V2TXLivePusher` . The UIs for screen recording operations in the `SDK API-Example` project are as shown below:



Tencent Cloud's proprietary RTC protocol features lower latency and better adaptability to poor network conditions when compared with traditional live streaming protocols, and can be used worldwide. For details, visit https://intl.cloud.tencent.com/document/product/

Everything on your screen, including notifications, will be recorded. Enable Do Not Disturb to prevent unexpected notifications.

Restrictions

The screen recording feature is available only on iOS 11.0 or later. This document describes how to use ReplayKit 2 on iOS 11 to push streams from the screen. The parts about the use of the SDK also apply to other custom stream push scenarios. For more information, see the code sample in the `TXReplayKit_Screen` folder of the demo.

Screen recording is a new feature on iOS 10. In addition to using ReplayKit to record video from the screen, which is possible on iOS 9, with iOS 10, users can also stream live video from the screen. For more information, see Go Live with ReplayKit. On iOS 11, Apple made ReplayKit more usable and more universally applicable and launched ReplayKit 2, going from supporting ReplayKit alone to allowing the recording of the entire screen. Therefore, we recommend you use ReplayKit 2 on iOS 11 to enable the screen recording feature. Screen recording relies on extensions, which operate as independent processes. However, to ensure system smoothness, iOS allocates limited resources to extensions and may kill extensions with high memory usage. Given this, Tencent Cloud has further reduced the memory usage of the LiteAV SDK while retaining its high streaming quality and low latency to ensure the stability of extensions.

Sample Code

Tencent Cloud offers an easy-to-understand API example project to help you quickly learn how to use different APIs.

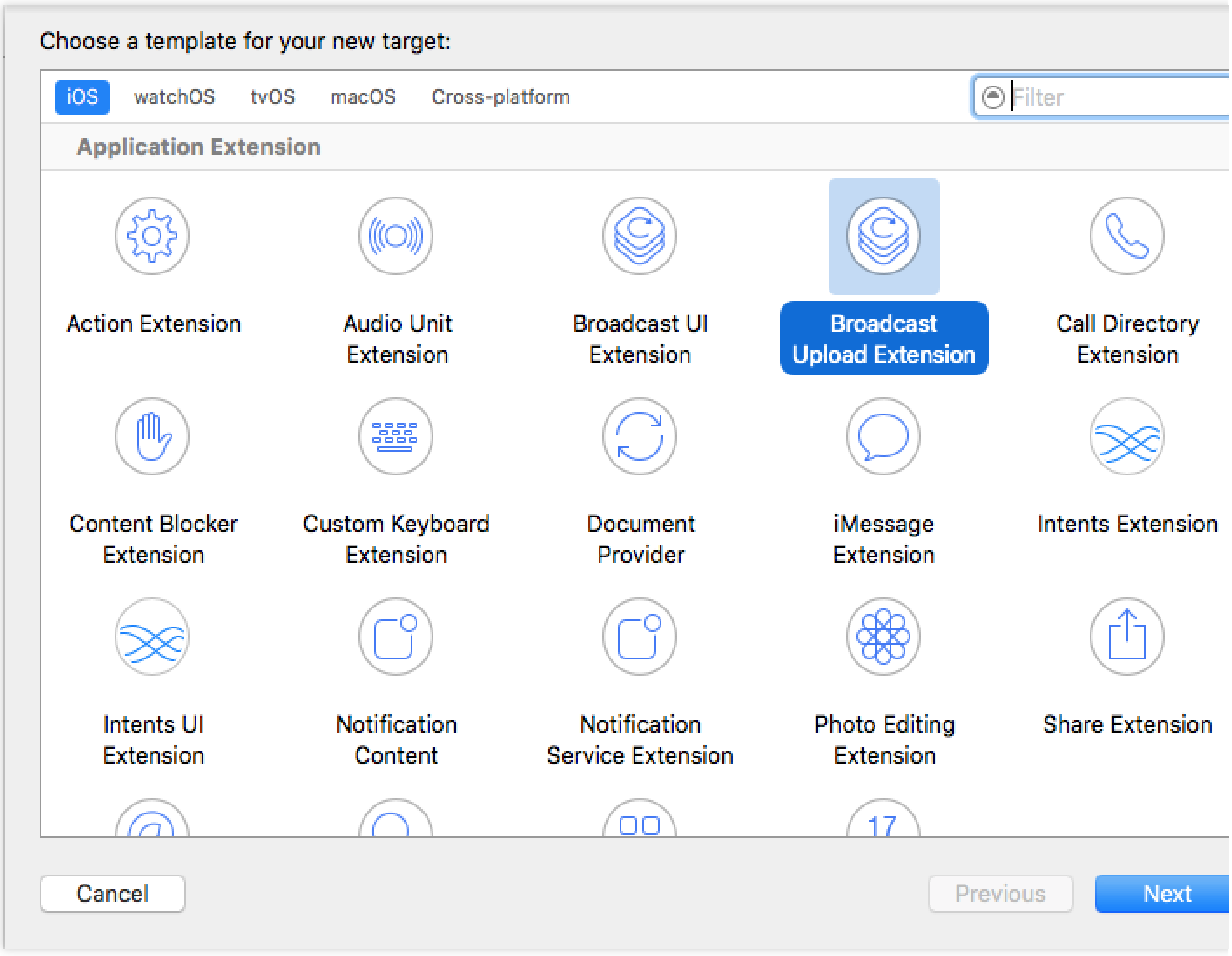
Platform	GitHub Address
iOS	LivePushScreenViewController.m
Android	LivePushScreenActivity.java
Flutter	live_screen_push.dart

Xcode

Xcode 9 or above is required, and your iPhone must be updated to iOS 11 or above. Screen recording is not supported on emulators.

Create a broadcast upload extension

Open your project with Xcode and select **New** > **Target...** > **Broadcast Upload Extension**, as shown below.



Enter a product name and click **Finish**. A new directory with the product name entered will appear in your project. Under the directory, there is an automatically generated `SampleHandler` class, which is responsible for screen recording operations.

SDK Integration

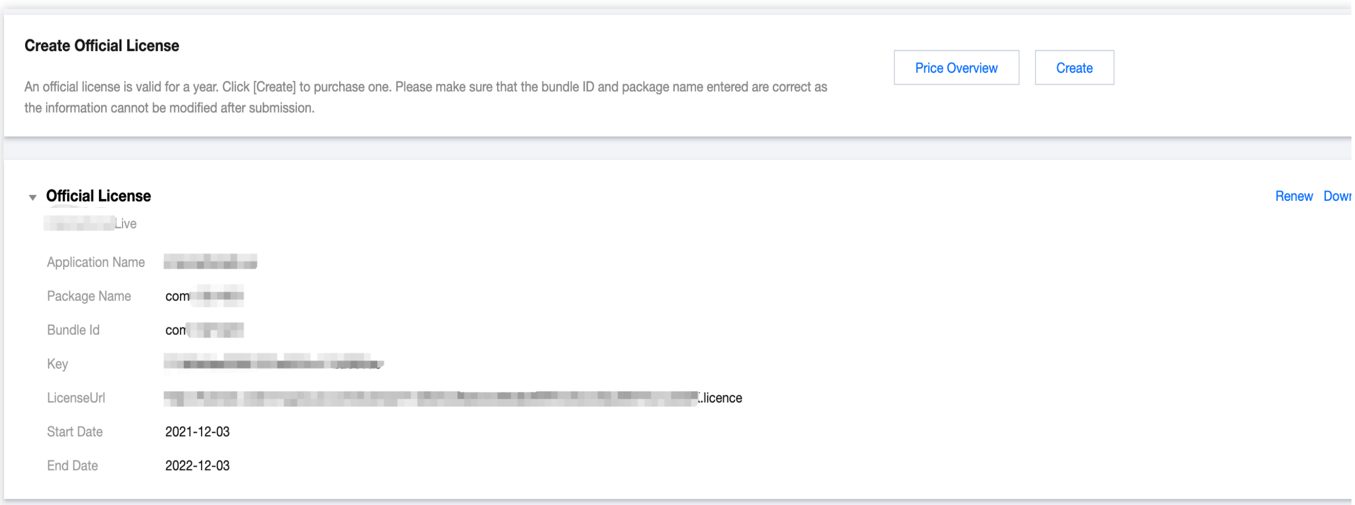
1. Download the SDK

Download the SDK and follow the instructions in [SDK Integration](https://www.tencentcloud.com/document/product/1071/38155 to integrate the SDK into your application.

2. Configure a license for the SDK

1. Get the license:

If you have the required license, get the license URL and key in the CSS console.


Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

Live

Application Name	[illegible]
Package Name	com
Bundle Id	con
Key	[illegible]
LicenseUrl	.licence
Start Date	2021-12-03
End Date	2022-12-03

If you don't have the required license, apply for a license as instructed in New License and Renewal.

2. Before your application calls features of LiteAVSDK, complete the following configuration (preferably in `- [AppDelegate application:didFinishLaunchingWithOptions:]`):

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<The license URL obtained>";
    NSString * const licenceKey = @"<The key obtained>";

     // `V2TXLivePremier` is in the `V2TXLivePremier.h` header file.
    [V2TXLivePremier setLicence:licenceURL key:licenceKey];
    [V2TXLivePremier setObserver:self];
    NSLog(@"SDK Version = %@", [V2TXLivePremier getSDKVersionStr]);
    return YES;
}

#pragma mark - V2TXLivePremierObserver
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
}
```

Note:

`BundleId` configured in the license must be the same as that of the application; otherwise, stream push will fail.

3. Initialize the `V2TXLivePusher` component

Create a `V2TXLivePusher` object and specify `V2TXLiveMode`.

```
// Set the live streaming protocol to RTMP, which doesn't support mic connect.
V2TXLivePusher *pusher = [[V2TXLivePusher alloc] initWithLiveMode:V2TXLiveMode_RTMP
```

4. Configure `RPBroadcastSampleHandler`

To use screen recording, you need to use a subclass of the system API `RPBroadcastSampleHandler` to get the screen audio/video data. Here, add the following code to the custom subclass SampleHandler.m to implement screen recording:

```
#import "SampleHandler.h"
@import TXLiteAVSDK_ReplayKitExt;

@implementation SampleHandler
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject *> *)setupI
    // The application group ID. The ID must match `startScreenCapture` of `V2TXLiv
    [[TXReplayKitExt sharedInstance] setupWithAppGroup:APPGROUP delegate:self];
}

- (void)broadcastPaused {
    // User has requested to pause the broadcast. Samples will stop being delivered
    [[TXReplayKitExt sharedInstance] broadcastPaused];
}

- (void)broadcastResumed {
    // User has requested to resume the broadcast. Samples delivery will resume.
    [[TXReplayKitExt sharedInstance] broadcastResumed];
}

- (void)broadcastFinished {
    // User has requested to finish the broadcast.
    [[TXReplayKitExt sharedInstance] broadcastFinished];
}

- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:(RPSampleBuffe
    [[TXReplayKitExt sharedInstance] sendSampleBuffer:sampleBuffer withType:sampleB
}

#pragma mark - TXReplayKitExtDelegate
- (void)broadcastFinished:(TXReplayKitExt *)broadcast reason:(TXReplayKitExtReason)
{
    NSString *tip = @"";
    switch (reason) {
        case TXReplayKitExtReasonRequestedByMain:
            tip = NSLocalizedString(@"MLVB-API-Example.liveStop", "");
            break;
        case TXReplayKitExtReasonDisconnected:
            tip = NSLocalizedString(@"MLVB-API-Example.appReset", "");
            break;
        case TXReplayKitExtReasonVersionMismatch:
            tip = NSLocalizedString(@"MLVB-API-Example.sdkError", "");
            break;
    }

    NSError *error = [NSError errorWithDomain:NSStringFromClass(self.class)
                                         code:0
                                     userInfo:@{
                                         NSLocalizedFailureReasonErrorKey:tip
                                     }];
    [self finishBroadcastWithError:error];
}

@end
```

In the implementation class of `RPBroadcastSampleHandler`, you need to call the corresponding method in `TXReplayKitExt` to set the screen recording information and handle screen recording events.

The **broadcastFinished** callback can be used to get the reason that screen recording is stopped and can be used to prompt the user to take action.

For details about communication between the extension and host app, see Communication and Data Transfer Between Extensions and Host Apps.

5. Enable screen recording and stream push

Call startScreenCapture to start screen recording and then call the startPush API in `V2TXLivePusher` to start pushing the stream.

Note:

If you select the `RTMP` protocol for stream push in step 3, you can generate a stream push address as instructed in Quick URL Generation.

```
// The application group ID for data sharing between the host application and Broad
[livePusher startScreenCapture:@"group.com.xxx"];
[livePusher startMicrophone];

// Pass in the corresponding URL based on the stream push protocol to start stream
NSString * const url = @"rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxx
V2TXLiveCode code = [livePusher startPush:url];
if (code != V2TXLIVE_OK) {
    // Check the error code
}
```

Why is `V2TXLIVE_ERROR_INVALID_LICENSE` returned?

If the startPush API returns V2TXLIVE_ERROR_INVALID_LICENSE, it means your license verification failed. Check the error code and error message in the onLicenceLoaded callback in Step 2. Configure a license for the SDK.

On iOS 11 or later, you can enable screen recording only by pulling down the status bar and pressing and holding the screen recording button.

On iOS 12 or later, you can use `RPSystemBroadcastPickerView` to pop up the screen recording selection page as instructed in TRTCBroadcastExtensionLauncher.m.

6. Set landscape stream push and the resolution

Call setVideoQuality to set the resolution (you can select from a number of resolutions) and orientation for stream publishing. Below is an example:

```
    BOOL landScape; // `YES`: Landscape; `NO`: Portrait.
    V2TXLiveVideoEncoderParam *videoParam = [[V2TXLiveVideoEncoderParam alloc] init
    videoParam.videoResolution = V2TXLiveVideoResolution960x540;
    videoParam.videoResolutionMode = landScape ? V2TXLiveVideoResolutionModeLandsca
    [livePusher setVideoQuality:videoParam];
```

7. Set the logo watermark

Call setWatermark in `V2TXLivePusher` to add a watermark to videos output by the SDK. The position of the watermark is determined by the `(x, y, scale)` parameter passed in.

The watermark image must be in PNG rather than JPG format. The former carries opacity information, which allows the SDK to better address the image aliasing issue (changing the extension of a JPG image to PNG won't work).

The `(x, y, scale)` parameter specifies the normalized coordinates of the watermark relative to the resolution of the published video. For example, if the resolution of the published video is 540 x 960, and `(x, y, scale)` is set to `(0.1, 0.1, 0.1)` , the actual pixel coordinates of the watermark will be (540 x 0.1, 960 x 0.1). The width of the watermark will be the video width x 0.1, and the height will be scaled automatically.

```
// Set the video watermark
[livePusher setWatermark:image x:0 y:0 scale:1];
```

8. Stop stream push

As there can be only one `V2TXLivePusher` object running at a time, make sure that you release all the resources when stopping publishing.

```
// Stop stream push
[livePusher stopScreenCapture];
[livePusher stopPush];
```

Event Handling

1. Listening for events

The SDK listens for publishing events and errors via the V2TXLivePusherObserver delegate. See V2TXLiveCode for a detailed list of events and error codes.

2. Errors

An error indicates that the SDK encountered a serious problem that made it impossible for stream publishing to continue.

Event ID	Code	Description
V2TXLIVE_ERROR_FAILED	-1	A common unclassified error occurred.
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	An invalid parameter was passed in during API calling.
V2TXLIVE_ERROR_REFUSED	-3	The API call was rejected.
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	The API cannot be called.
V2TXLIVE_ERROR_INVALID_LICENSE	-5	Failed to call the API due to invalid license.
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	The server request timed out.
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	The server could not handle your request.

3. Warnings

A warning indicates that the SDK encountered a problem whose severity level is warning. Warning events trigger tentative protection or recovery logic and can often be resolved.

Event ID	Code	Description
V2TXLIVE_WARNING_NETWORK_BUSY	1101	Bad network connection: Data upload blocked due to limited upstream bandwidth.
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	Stuttering during video playback.
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	Failed to turn the camera on.
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	The camera is occupied. Try a different camera.
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	No access to the camera. This usually occurs on mobile devices and may be because the user denied the access.

V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	Failed to turn the mic on.
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	The mic is occupied. This occurs when, for example, the user is currently having a call on the mobile device.
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	No access to the mic. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	The system does not support screen sharing.
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	Failed to start screen recording. If this occurs on a mobile device, it may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	Screen recording was stopped by the system

Appendix: Communication and Data Transfer Between Extensions and Host Applications

ReplayKit2 invokes only the broadcast upload extension during screen sharing. The extension does not support UI operations and cannot implement complicated business logic. Therefore, the host app is often responsible for implementing business logic such as authentication, while the extension focuses on recording the screen and publishing the audio and video data captured. This makes it necessary to communicate and transfer data between the extension and host app.

1. Sending local notifications

Users should be informed of the status of the extension. For example, in cases where the host app is not started, you can send a local notification asking users to interact with the host app. Below is an example of sending a notification to users when the broadcast upload extension is started.

```
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject *> *)setupI
    [self sendLocalNotificationToHostAppWithTitle:@"Tencent Cloud Screen Sharing" m
}
```

```
- (void)sendLocalNotificationToHostAppWithTitle:(NSString*)title msg:(NSString*)msg
{
    UNUserNotificationCenter* center = [UNUserNotificationCenter currentNotificatio

    UNMutableNotificationContent* content = [[UNMutableNotificationContent alloc] i
    content.title = [NSString localizedUserNotificationStringForKey:title arguments
    content.body = [NSString localizedUserNotificationStringForKey:msg  arguments:n
    content.sound = [UNNotificationSound defaultSound];
    content.userInfo = userInfo;

    // Schedule a local notification to be sent at the specified time
    UNTimeIntervalNotificationTrigger* trigger = [UNTimeIntervalNotificationTrigger
                                              triggerWithTimeInterval:0.1f repe

    UNNotificationRequest* request = [UNNotificationRequest requestWithIdentifier:@
                                                                        content:c

    // Add operation after the notification is sent
    [center addNotificationRequest:request withCompletionHandler:^(NSError * _Nulla

    }];
}
```

You can use this notification to ask the user to return to the host app to configure publishing information and start stream publishing.

2. Sending notifications between processes via CFNotificationCenter

The extension and host app may also need to interact with each other in real time, which cannot be achieved through local notifications because with local notifications, code is triggered only after users tap the banner. Neither can it be implemented via NSNotificationCenter because NSNotificationCenter does not allow communication between processes. To send notifications between processes, you will need CFNotificationCenter, but instead of using the `userInfo` field for data transfer, you must configure an app group and use `NSUserDefault` for data transfer. For example, after getting the publishing URL, the host app can notify the broadcast upload extension via CFNotificationCenter that stream publishing can start. You may also use the clipboard, but delayed rendering is needed as the clipboard sometimes fails to transfer data between processes in real time.

```
CFNotificationCenterPostNotification(CFNotificationCenterGetDarwinNotifyCenter(),
                                    kDarvinNotificationNamePushStart,
                                    NULL,
                                    nil,
                                    YES);
```

The extension can start publishing streams after receiving this notification. As the notification is at the CF layer, to facilitate operations, it needs to be sent to the Cocoa layer via NSNotificationCenter.

```
    CFNotificationCenterAddObserver(CFNotificationCenterGetDarwinNotifyCenter(),
                                    (__bridge const void *)(self),
                                    onDarwinReplayKit2PushStart,
                                    kDarvinNotificationNamePushStart,
                                    NULL,
                                    CFNotificationSuspensionBehaviorDeliverImmediat

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(handl


static void onDarwinReplayKit2PushStart(CFNotificationCenterRef center,
                                        void *observer, CFStringRef name,
                                        const void *object, CFDictionaryRef
                                        userInfo)
{
// Send to the Cocoa layer
    [[NSNotificationCenter defaultCenter] postNotificationName:@"Cocoa_ReplayKit2_P
}

- (void)handleReplayKit2PushStartNotification:(NSNotification*)noti
{
// Get the data to be transferred by the host app via NSUserDefault or the clipboar
//    NSUserDefaults *defaults = [[NSUserDefaults alloc] initWithSuiteName:kReplayK

    UIPasteboard* pb = [UIPasteboard generalPasteboard];
    NSDictionary* defaults = [self jsonData2Dictionary:pb.string];

    s_rtmpUrl = [defaults objectForKey:kReplayKit2PushUrlKey];
    s_resolution = [defaults objectForKey:kReplayKit2ResolutionKey];
    if (s_resolution.length < 1) {
        s_resolution = kResolutionHD;
    }
    NSString* rotate = [defaults objectForKey:kReplayKit2RotateKey];
    if ([rotate isEqualToString:kReplayKit2Portrait]) {
        s_landScape = NO;
    }
    else {
        s_landScape = YES;
    }
    [self start];
}
```

FAQs

ReplayKit2 is a new framework introduced by Apple in iOS 11, for which relatively few official documents have been released. The framework is still being improved, and problems have been found. See below for some common questions you may have when using ReplayKit2.

1. **When does screen recording stop automatically?**

Screen recording stops automatically when the screen locks or there is an incoming call. At such times, the `broadcastFinished` function in `SampleHandler` will be invoked, and you can send a notification to users about the interruption.

2. **Why does screen recording stop sometimes during screen sharing?**

The problem usually occurs after landscape/portrait mode switch if the resolution for stream publishing is set high. The broadcast upload extension is allocated a memory of only 50 MB and will be killed if its memory usage exceeds the limit. Given this, we recommend that you set the resolution to 720p or lower.

3. **Why are images streamed from the screen of iPhone X distorted?**

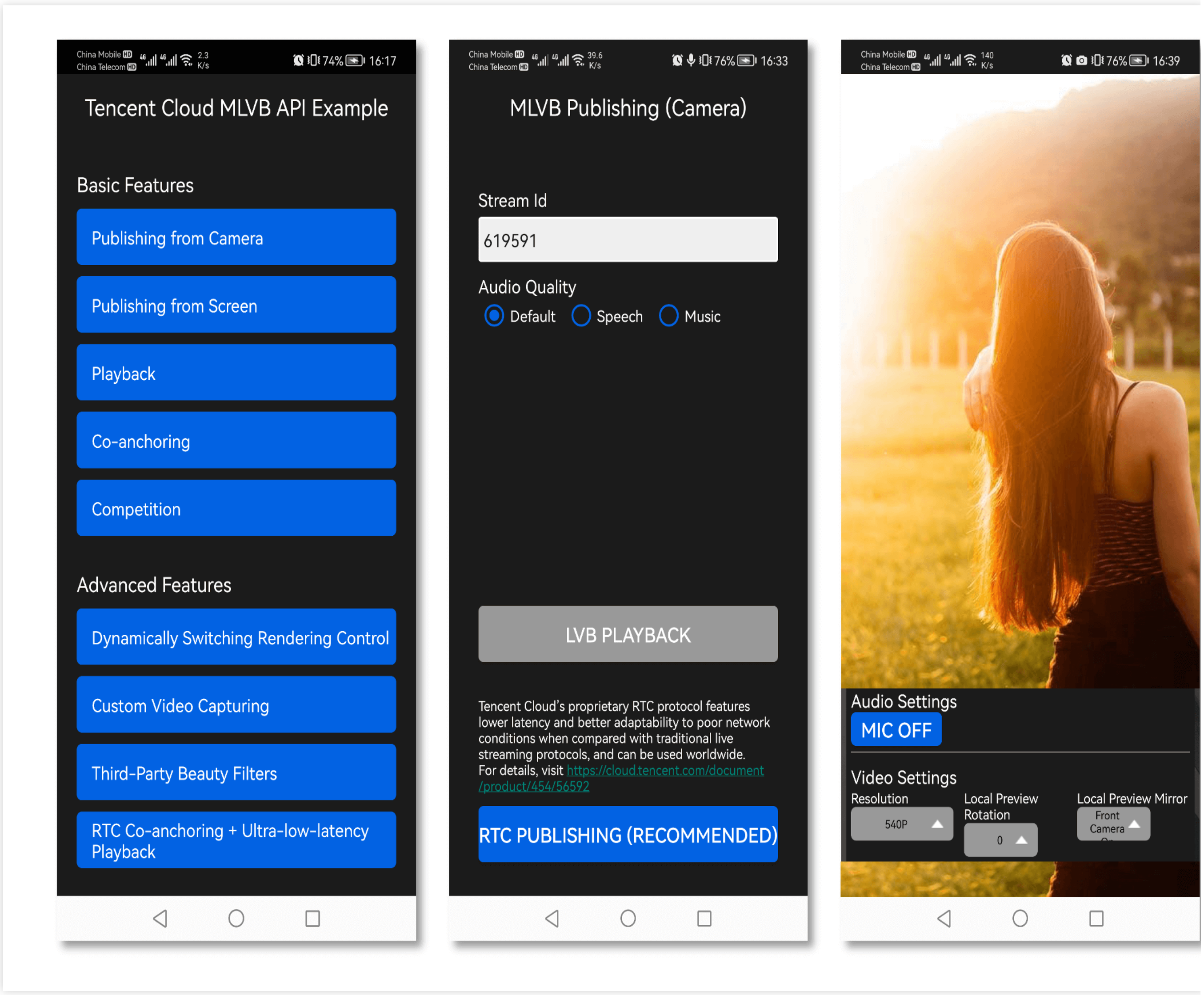
iPhone X has a notch at the top of the screen, so video captured from the screen is not in the aspect ratio of 16:9. If you set the output resolution for stream publishing to 16:9, for example, to HD (960 × 540), the images published will be slightly distorted because their original aspect ratio is not 16:9. We recommend that you set the resolution according to your screen size. Besides, if you play video streamed from the screen of iPhone X in aspect fit mode, the video may have black bars, and if you play it in aspect fill mode, the video may be cropped.

Android
Publishing from Camera

Last updated：2024-01-13 15:49:41

Overview

Publishing from camera refers to the process of collecting video and audio data from the mobile phone's camera and mic, encoding the data, and publishing it to cloud-based live streaming platforms. Tencent Cloud's LiteAVSDK provides the camera publishing capability via `V2TXLivePusher` .The following is the relevant GUI that demonstrating camera push stream in the SDK API-Example project:



Stream Id: 619591

Audio Quality: [✓ Default] Speech Music

Tencent Cloud's proprietary RTC protocol features lower latency and better adaptability to poor network conditions when compared with traditional live streaming protocols, and can be used worldwide. For details, visit https://cloud.tencent.com/document/product/454/56592

Resolution: 540P

Local Preview Rotation: 0

Local Preview Mirror: Front Camera

Notes

Testing on real devices: The SDK uses a lot of audio and video APIs of the Android system, most of which cannot be used on emulators. Therefore, we recommend that you test your project on a real device.

Sample Code

Platform	GitHub Address	Key Class
iOS	Github	LivePushCameraViewController.m
Android	Github	LivePushCameraActivity.java
Flutter	Github	live_camera_push.dart

Integration

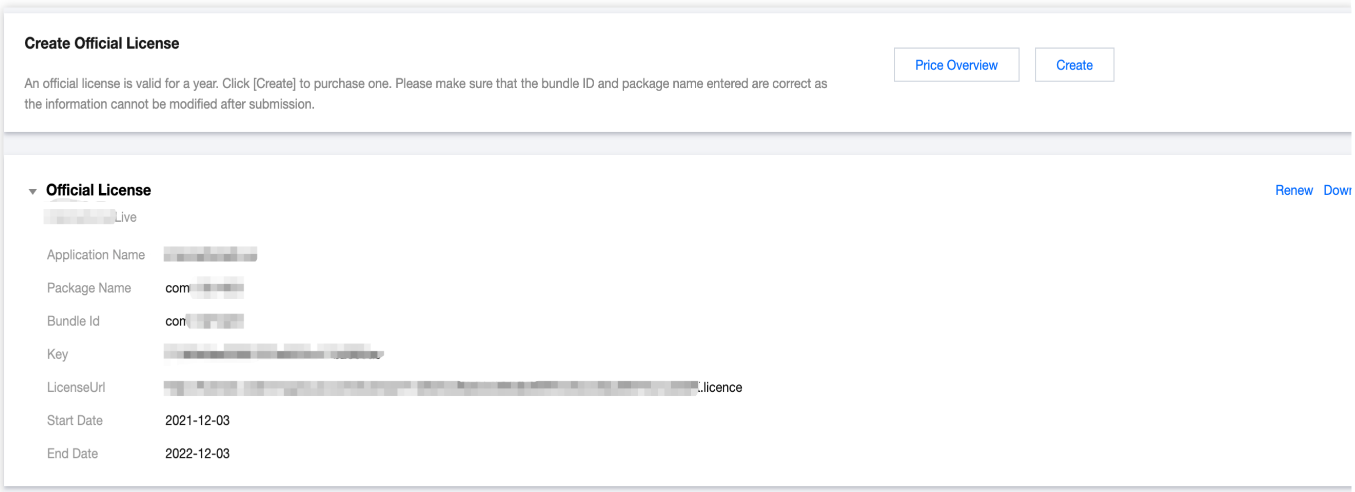
1. Download the SDK

Download the SDK and follow the instructions in SDK Integration to integrate the SDK into your application.

2. Configure License Authorization for SDK

1. Obtain license authorization：

If you have obtained the relevant license authorization, need to Get License URL and License Key in Cloud Live Console



Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

Application Name	
Package Name	com
Bundle Id	con
Key	
LicenseUrl	.licence
Start Date	2021-12-03
End Date	2022-12-03

If you have not yet obtained the license authorization, please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:

```
public class MApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // your licence url
String licenceKey = ""; // your licence key
V2TXLivePremier.setEnvironment("GDPR");  // set your environment
V2TXLivePremier.setLicence(this, licenceURL, licenceKey);
V2TXLivePremier.setObserver(new V2TXLivePremierObserver() {
        @Override
        public void onLicenceLoaded(int result, String reason) {
            Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reason);
        }
    });
}
```

3. Initialize the `V2TXLivePusher` component

Create a `V2TXLivePusher` object, which will be responsible for publishing operations.

```
// Specify the corresponding live broadcast protocol as RTMP, which does not suppor
V2TXLivePusher mLivePusher = new V2TXLivePusherImpl(this, V2TXLiveDef.V2TXLiveMode.
```

4. Enable camera preview

Before enabling camera preview, you must first provide the SDK with a `TXCloudVideoView` object to display video images. Given that `TXCloudVideoView` is inherited from `FrameLayout` in Android, you can:

1. Add a video rendering control in the XML file:

```
<com.tencent.rtmp.ui.TXCloudVideoView
           android:id="@+id/pusher_tx_cloud_view"
           android:layout_width="match_parent"
           android:layout_height="match_parent" />
```

2. Call `startCamera` in `V2TXLivePusher` to enable camera preview for your mobile phone.

```
// Enable preview for the local camera
TXCloudVideoView mPusherView = (TXCloudVideoView) findViewById(R.id.pusher_tx_cloud
mLivePusher.setRenderView(mPusherView);
mLivePusher.startCamera(true);
mLivePusher.startMicrophone();
```

5. Start and stop publishing

1. After calling `startCamera` to enable camera preview, you can call the startPush API in `V2TXLivePusher` to start publishing.

Note

if you choose `RTMP` protocol to push in Step3, the generate of the push URL, please refer to RTMP URL.

```
//This URL does not support co-anchoring. The stream is published to a live streami
String url = "rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx";
int ret = mLivePusher.startPush(url);
if (ret == V2TXLIVE_ERROR_INVALID_LICENSE) {
 Log.i(TAG, "startRTMPPush: license verification failed");
}
```

Note

Reason for returning V2TXLIVE_ERROR_INVALID_LICENSE?

If the startPush interface returns V2TXLIVE_ERROR_INVALID_LICENSE, it means that your license verification failed, please check the url and key set in Step 2: Configure the SDK for license authorization.

2. Call stopPush in `V2TXLivePusher` to stop publishing streams

```
//Stop publishing
mLivePusher.stopPush();
```

6. Publish audio-only streams

If your live streaming scenarios involve audio only, you can skip Step 4 or call `stopCamera` before `startPush` .

```
mLivePusher.startMicrophone();
// The push stream can be started by passing in the corresponding URL according to
String url = "rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx";
int ret = mLivePusher.startPush(url);
```

Note

If you publish audio-only streams but no streams can be pulled from an RTMP, FLV, or HLS playback URL, there is a problem with your line configuration, please submit a ticket for help.

7. Set video quality

Call setVideoQuality in `V2TXLivePusher` to set the quality of videos watched by audience. The encoding parameters set determine the quality of videos presented to audience. The local video watched by the host is the original HD version that has not been encoded or compressed, and is therefore not affected by the settings. For details, please see Setting Video Quality.

8. Set the beauty filter style and skin brightening and rosy skin effects

Call getBeautyManager in `V2TXLivePusher` to get a `TXBeautyManager` instance to set beauty filters.

Beauty filter style

The SDK has three built-in beauty filter algorithms, each corresponding to a beauty filter style. Choose one that best fits your product positioning. For the definitions, see `TXLiveConstants.java` .

Beauty Filter Style	Description
BEAUTY_STYLE_SMOOTH	The smooth style, which features more obvious skin smoothing effects and is suitable for live showrooms
BEAUTY_STYLE_NATURE	The natural style, which retains more facial details and is more natural
BEAUTY_STYLE_PITU	The Pitu style, which uses the beauty filter algorithm developed by YouTu Lab. Its effect is between the smooth style and the natural style, that is, it retains more skin details than the smooth style and delivers more obvious skin smoothing effects than the natural style.

You can call the `setBeautyStyle` API of `TXBeautyManager` to set the beauty filter style.

Item	Configuration	Description
Beauty filter strength	Via the setBeautyLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the more obvious the effect.
Skin brightening filter strength	Via the setWhitenessLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the more obvious the effect.
Rosy skin filter strength	Via the setRuddyLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the more obvious the effect.

9. Set color filters

Call getBeautyManager in `V2TXLivePusher` to get a TXBeautyManager instance to set color filters.
Call the `setFilter` API in `TXBeautyManager` to set color filters. Color filters are a technology that adjusts the color tone of sections of an image. For example, it may lighten the yellow sections of an image to achieve the effect of skin brightening, or add warm tones to a video to give it a refreshing and soft boost.
Call the `setFilterStrength` API in `TXBeautyManager` to set the strength of a color filter. The higher the strength, the more obvious the effect.
Based on our experience of operating Mobile QQ and Now Live, it’s not enough to use only the `setBeautyStyle` API in `TXBeautyManager` to set the beauty filter style. The `setBeautyStyle` API must be used together with

`setFilter` to produce richer effects. Given this, our designers have developed 17 built-in color filters for you to choose from.

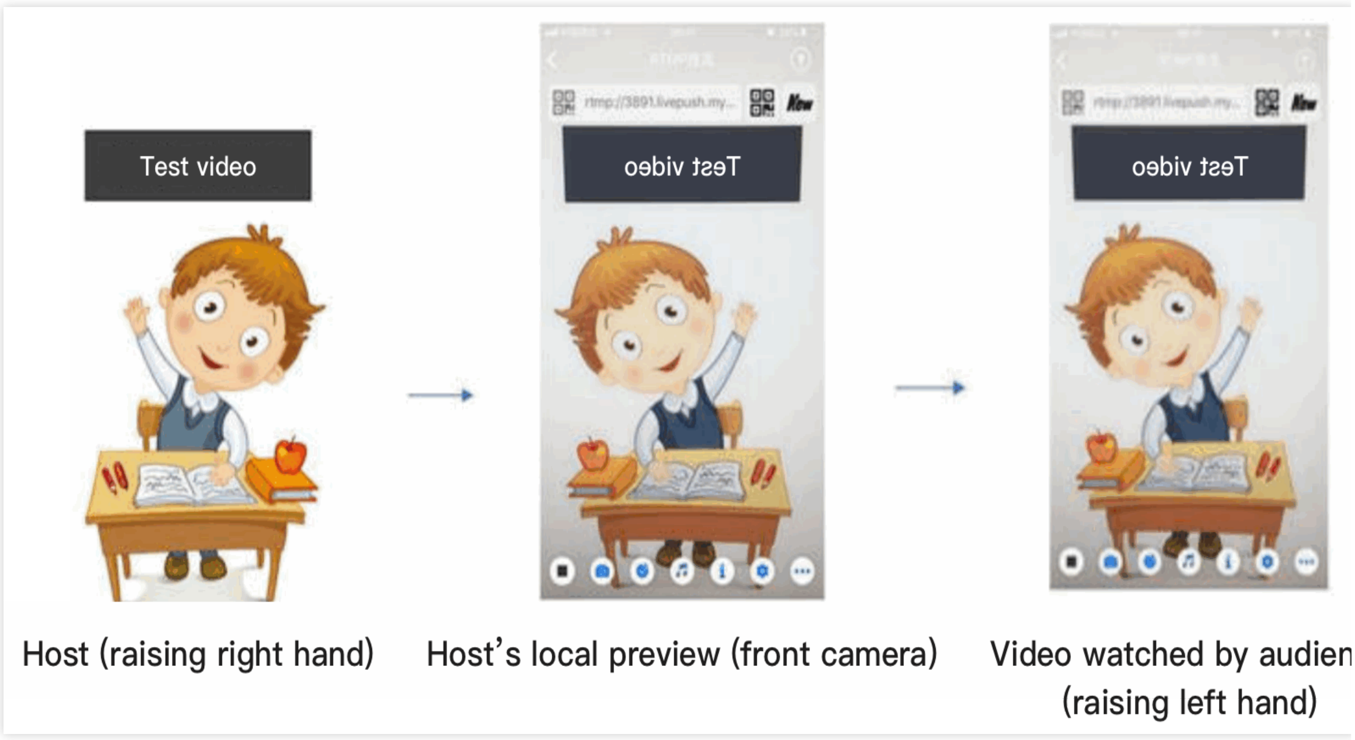
```
// Select the desired color filter file. The filter file can be obtained from the r
Bitmap filterBmp = decodeResource(getResources(), R.drawable.tuibeauty_filter_biaoz
mLivePusher.getBeautyManager().setFilter(filterBmp);
mLivePusher.getBeautyManager().setFilterStrength(0.5f);
```

10. Manage devices

`V2TXLivePusher` provides a series of APIs for the control of devices. You can call `getDeviceManager` to get a `TXDeviceManager` instance for device management. For detailed instructions, please see TXDeviceManager API.

11. Set the video mirroring effect for audience

Call setEncoderMirror in `V2TXLivePusher` to set the camera mirror mode, which affects the way video images are presented to audience. By default, the local video watched by the host is flipped when the front camera is used, which creates the same effect as a mirror does. The video watched by audience is the same as that watched by the host, as shown below.



12. Publish streams in landscape mode

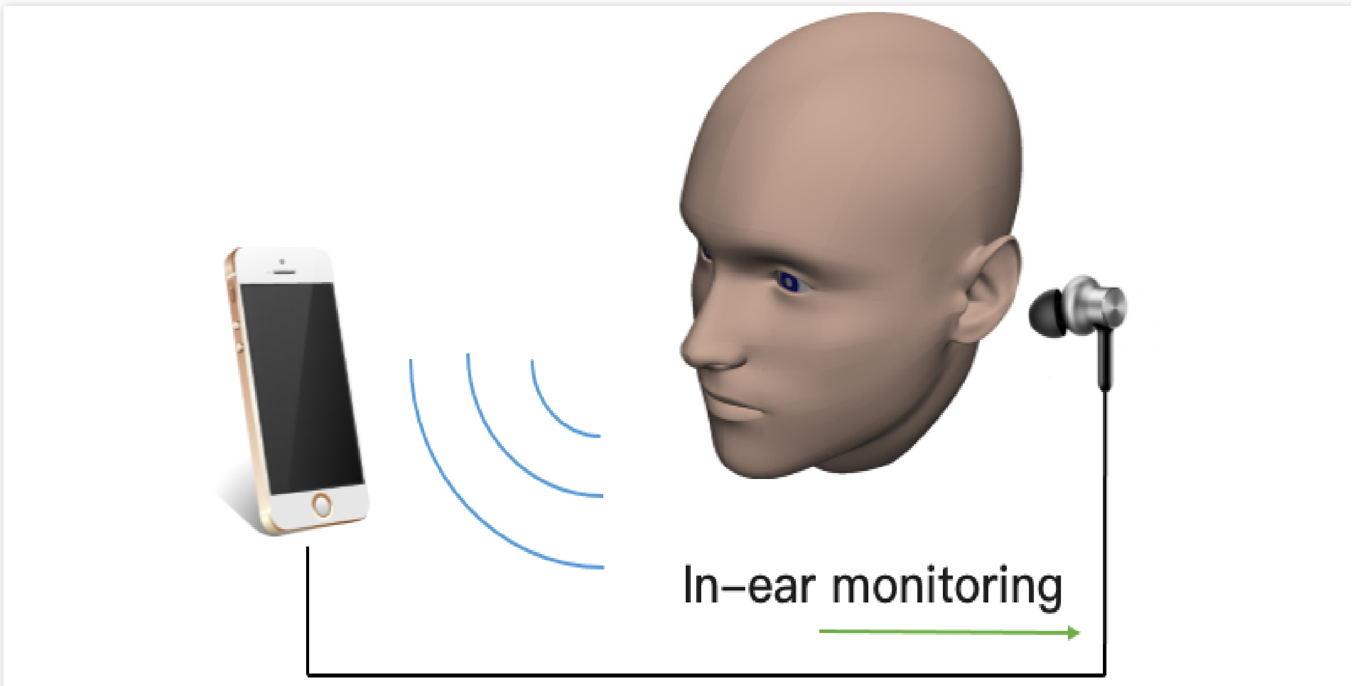
In most cases, hosts stream while holding their phones vertically, and audience watch videos in portrait resolutions (e.g., 540 × 960). However, there are also cases where hosts hold phones horizontally, and ideally, audience should

watch videos in landscape resolutions (960 × 540).
By default, `V2TXLivePusher` outputs videos in portrait resolutions. You can publish landscape-mode videos to audience by modifying a parameter of the `setVideoQuality` API.

```
V2TXLiveDef.V2TXLiveVideoEncoderParam param = new V2TXLiveDef.V2TXLiveVideoEncoderP
param.videoResolutionMode = isLandscape ? V2TXLiveVideoResolutionModeLandscape : V2
mLivePusher.setVideoQuality(param);
```

13. Set audio effects

Call `getAudioEffectManager` in `V2TXLivePusher` to get a `TXAudioEffectManager` instance, which can be used to mix background music and set in-ear monitoring, reverb, and other audio effects. Background music mixing means mixing into the published stream the music played by the host's phone so that audience can also hear the music.
Call the `enableVoiceEarMonitor` API in TXAudioEffectManager to enable in-ear monitoring, which allows hosts to hear their vocals in earphones when they sing.
Call the `setVoiceReverbType` API in `TXAudioEffectManager` to add reverb effects such as karaoke, hall, husky, and metal. The effects are applied to the videos watched by audience.
Call the `setVoiceChangerType` API in `TXAudioEffectManager` to add voice changing effects such as little girl and middle-aged man to enrich host-audience interaction. The effects are applied to the videos watched by audience.



In–ear Monitoring Illustration

Note

For detailed instructions, please see TXAudioEffectManager API.

14. Set watermarks

Call setWatermark in `V2TXLivePusher` to add a watermark to videos output by the SDK. The position of the watermark is determined by the `(x, y, scale)` parameter passed in.

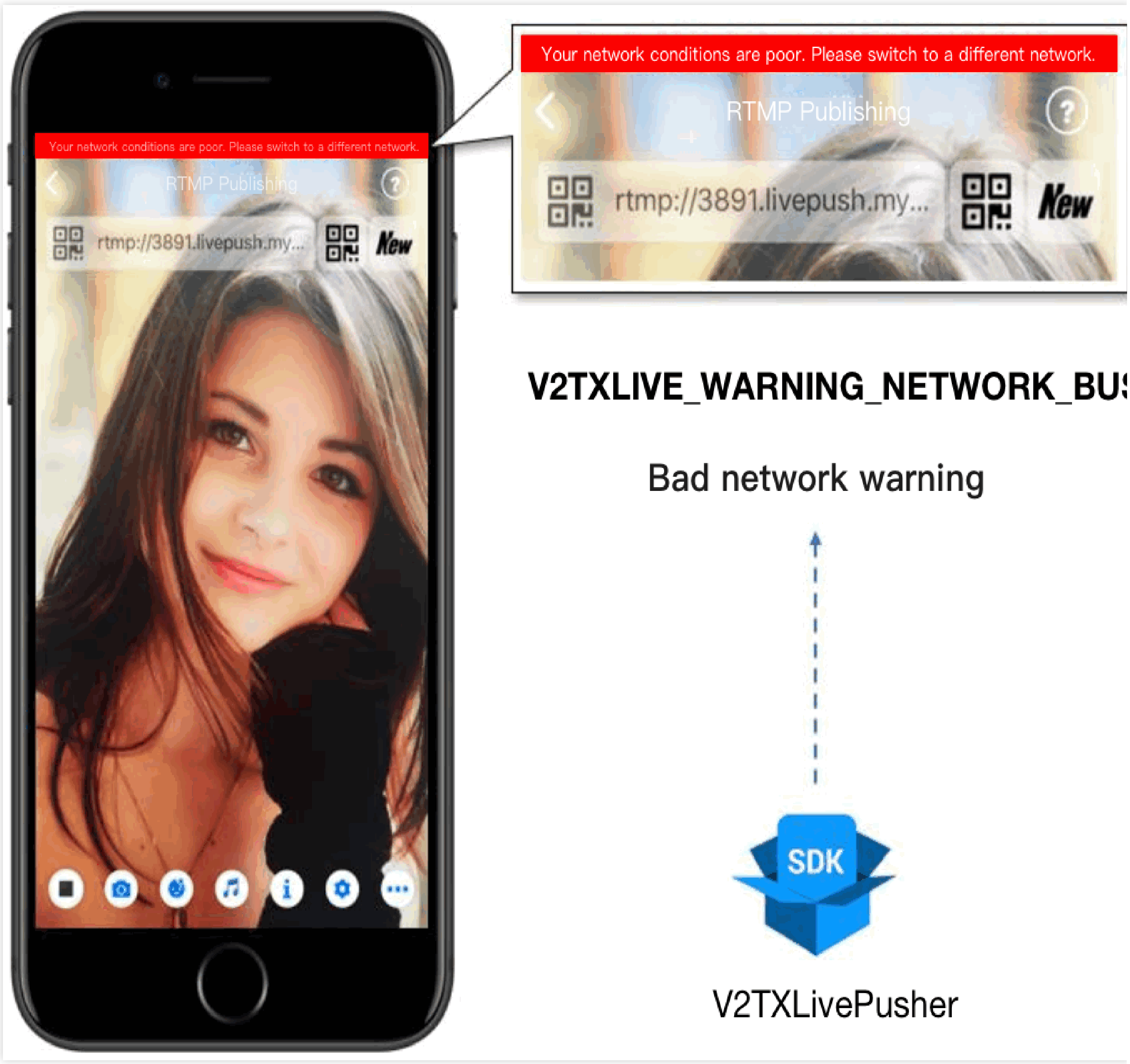
The watermark image must be in PNG rather than JPG format. The former carries opacity information, which allows the SDK to better address the image aliasing issue (changing the extension of a JPG image to PNG won't work).

The `(x, y, scale)` parameter specifies the normalized coordinates of the watermark relative to the resolution of the published video. For example, if the resolution of the published video is 540 x 960, and `(x, y, scale)` is set to `(0.1, 0.1, 0.1)`, the actual pixel coordinates of the watermark will be (540 x 0.1, 960 x 0.1). The width of the watermark image will be the video width x 0.1, and the height will be scaled automatically.

```
// Set a video watermark
mLivePusher.setWatermark(BitmapFactory.decodeResource(getResources(),R.drawable.wat
```

15. Inform hosts of poor network conditions

Hosts should be informed when their network conditions are bad and be prompted to check their network.


Your network conditions are poor. Please switch to a different network.
RTMP Publishing
rtmp://3891.livepush.my...
New
V2TXLIVE_WARNING_NETWORK_BUS
Bad network warning
SDK
V2TXLivePusher

You can capture the **V2TXLIVE_WARNING_NETWORK_BUSY** event using onWarning in `V2TXLivePusherObserver` . The event indicates poor network conditions for hosts, which result in stuttering for audience. When this event occurs, you can send a UI message about poor network conditions to hosts.

```
@Override
public void onWarning(int code, String msg, Bundle extraInfo) {
    if (code == V2TXLiveCode.V2TXLIVE_WARNING_NETWORK_BUSY) {
        showNetBusyTips(); // Show network tips
    }
}
```

16. Send SEI messages

Call the sendSeiMessage API in `V2TXLivePusher` to send SEI messages. SEI refers to the supplementary enhancement information of encoded video. It is not used most of the time, but you can insert custom information into SEI messages. The information will be forwarded to audience by live streaming CDNs. The applications for SEI messages include:
Live quiz: The publisher can use SEI messages to send questions to the audience. SEI can ensure synchronization among audio, video, and the questions.
Live showroom: The publisher can use SEI messages to display lyrics to the audience in real time. The effects are not affected by reduction in video encoding quality.
Online education: The publisher can use SEI messages to display pointers and sketches on slides to the audience in real time.
Custom data is inserted directly into video data and therefore cannot be too large in size (preferably several bytes). It's common to insert information such as custom timestamps.

```
//Sample code for Android
int payloadType = 5;
String msg = "test";
mTXLivePusher.sendSeiMessage(payloadType, msg.getBytes("UTF-8"));
```

Common open-source players or web players are incapable of parsing SEI messages. You must use `V2TXLivePlayer` , the built-in player of LiteAVSDK.
1. Configuration:

```
int payloadType = 5;
mTXLivePlayer.enableReceiveSeiMessage(true, payloadType)
```

2. If the video streams played by `V2TXLivePlayer` contain SEI messages, you will receive the messages via the `onReceiveSeiMessage` callback in `V2TXLivePlayerObserver` .

Event Handling

Listening for events

The SDK listens for publishing events and errors via the V2TXLivePusherObserver delegate. See V2TXLiveCode for a detailed list of events and error codes.

Errors

An error indicates that the SDK encountered a serious problem that made it impossible for stream publishing to continue.

Event ID	Code	Description
V2TXLIVE_ERROR_FAILED	-1	A common error not yet classified

V2TXLIVE_ERROR_INVALID_PARAMETER	-2	An invalid parameter was passed in during API calling.
V2TXLIVE_ERROR_REFUSED	-3	The API call was rejected.
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	The API cannot be called.
V2TXLIVE_ERROR_INVALID_LICENSE	-5	Failed to call the API due to invalid license.
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	The server request timed out.
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	The server could not handle your request.

Warnings

A warning indicates that the SDK encountered a problem whose severity level is warning. Warning events trigger tentative protection or recovery logic and can often be resolved.

Event ID	Code	Description
V2TXLIVE_WARNING_NETWORK_BUSY	1101	Bad network connection: data upload blocked due to limited upstream bandwidth.
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	Stuttering during video playback.
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	Failed to turn the camera on.
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	The camera is occupied. Try a different camera.
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	No access to the camera. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	Failed to turn the mic on.
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	The mic is occupied. This occurs when, for example, the user is having a call on the mobile device.
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	No access to the mic. This usually occurs on mobile

		devices and may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	The system does not support screen sharing.
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	Failed to start screen recording. If this occurs on a mobile device, it may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	Screen recording was stopped by the system.

Publishing from Screen

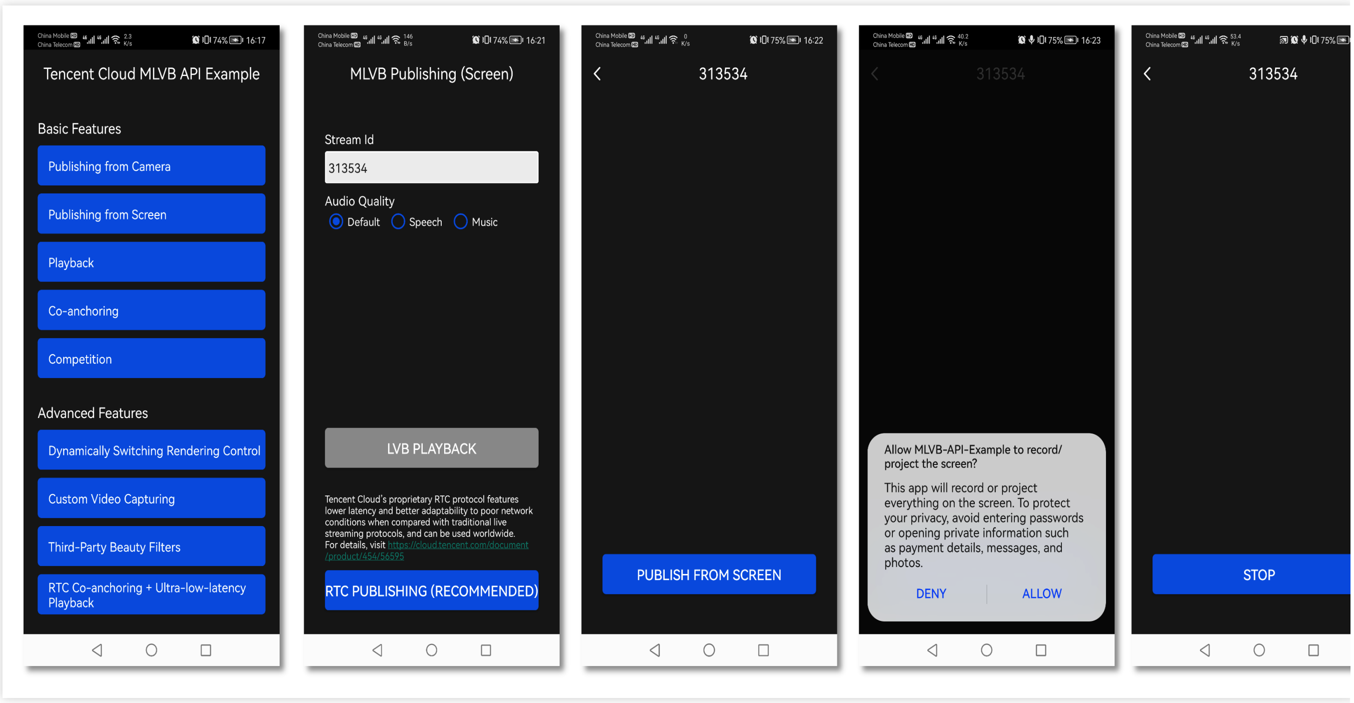
Last updated：2024-01-13 15:49:41

Overview

Publishing from screen means using hosts’ phones as the source for live streaming. It may be combined with local camera preview and is used in scenarios such as game streaming and mobile application demonstration. Tencent Cloud’s LiteAVSDK offers the screen sharing capability via `V2TXLivePusher` .

Note:

By adding a floating window to display the image of the local camera, you can include camera preview into the streams published from the screen.



Stream Id: 313534

Audio Quality: [✓ Default] Speech Music

Tencent Cloud’s proprietary RTC protocol features lower latency and better adaptability to poor network conditions when compared with traditional live streaming protocols, and can be used worldwide. For details, visit https://cloud.tencent.com/document/product/454/56595

Allow MLVB-API-Example to record/ project the screen?

This app will record or project everything on the screen. To protect your privacy, avoid entering passwords or opening private information such as payment details, messages, and photos.

Restrictions

Screen recording is supported in Android 5.0 and above.

Floating windows need to be enabled manually on some mobile phones and systems.

Sample Code

Platform	GitHub Address
iOS	LivePushScreenViewController.m
Android	LivePushScreenActivity.java
Flutter	live_screen_push.dart

Integration

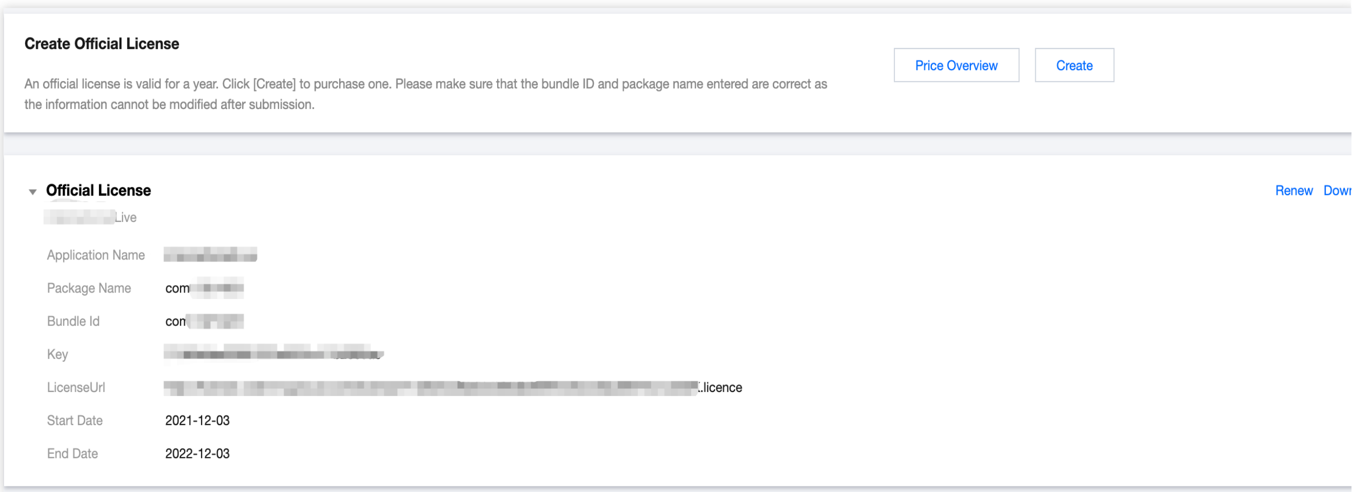
Step 1. Download SDK development kit

Download the SDK and follow the instructions in SDK Integration to integrate the SDK into your application.

Step 2. Configure License Authorization for SDK

1. Obtain license authorization：

If you have obtained the relevant license authorization， Need to Get License URL and License Key in Cloud Live Console



Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

[redacted]Live

Application Name	[redacted]
Package Name	com[redacted]
Bundle Id	con[redacted]
Key	[redacted]
LicenseUrl	[redacted].licence
Start Date	2021-12-03
End Date	2022-12-03

If you have not yet obtained the license authorization， Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:

```
public class MApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // your licence url
String licenceKey = ""; // your licence key
V2TXLivePremier.setEnvironment("GDPR");  // set your environment
V2TXLivePremier.setLicence(this, licenceURL, licenceKey);
V2TXLivePremier.setObserver(new V2TXLivePremierObserver() {
        @Override
        public void onLicenceLoaded(int result, String reason) {
            Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reason);
        }
    });
}
```

Note:

The packageName configured in the license must be the same as the application itself, otherwise the push stream will fail.

Step 3. Add Activity

Paste the following activity in the manifest file (no need to add it if it exists in the project code).

```
<activity
    android:name="com.tencent.rtmp.video.TXScreenCapture$TXScreenCaptureAssistantAc
    android:theme="@android:style/Theme.Translucent"/>
```

Step 4. Create a stream publishing object

Create a **V2TXLivePusher** object, which will be responsible for publishing operations.

```
// Specify the corresponding live broadcast protocol as RTMP, which does not suppor
V2TXLivePusher mLivePusher = new V2TXLivePusherImpl(this, V2TXLiveDef.V2TXLiveMode.
```

Step 5. Start stream publishing

Use startScreenCapture to start screen recording, and use V2TXLivePusher::startPush to pushing

Note:

if you choose `RTMP` protocol to push in Step4, The generate of the push URL, please refer to RTMP URL。

```
// The push stream can be started according to the push stream protocol. RTMP canno
String url = "rtmp://test.com/live/streamid?txSecret=xxxxx&txTime=xxxxxxxx";
mLivePusher.startMicrophone();
mLivePusher.startScreenCapture();
int ret = mLivePusher.startPush(url);
if (ret == V2TXLIVE_ERROR_INVALID_LICENSE) {
    Log.i(TAG, "startRTMPPush: license verification failed");
}
```

Note:

Reason for returning V2TXLIVE_ERROR_INVALID_LICENSE?

If the `startPush` interface returns `V2TXLIVE_ERROR_INVALID_LICENSE` , it means that your license verification failed, please check the url and key set in Step 2: Configure the SDK for license authorization.

startScreenCapture is used to start screen recording, which is a built-in feature of the Android system. For security reasons, before screen recording starts, Android will pop up a window asking users whether to start screen recording, and you should agree.

Step 6. Set watermarks

Call setWatermark in `V2TXLivePusher` to add a watermark to videos output by the SDK. The position of the watermark is determined by the `(x, y, scale)` parameter passed in.

The watermark image must be in PNG rather than JPG format. The former carries opacity information, which allows the SDK to better address the image aliasing issue (changing the extension of a JPG image to PNG won't work).

The `(x, y, scale)` parameter specifies the normalized coordinates of the watermark relative to the resolution of the published video. For example, if the resolution of the published video is 540 x 960, and `(x, y, scale)` is set to `(0.1, 0.1, 0.1)` , the actual pixel coordinates of the watermark will be (540 x 0.1, 960 x 0.1). The width of the watermark will be the video width x 0.1, and the height will be scaled automatically.

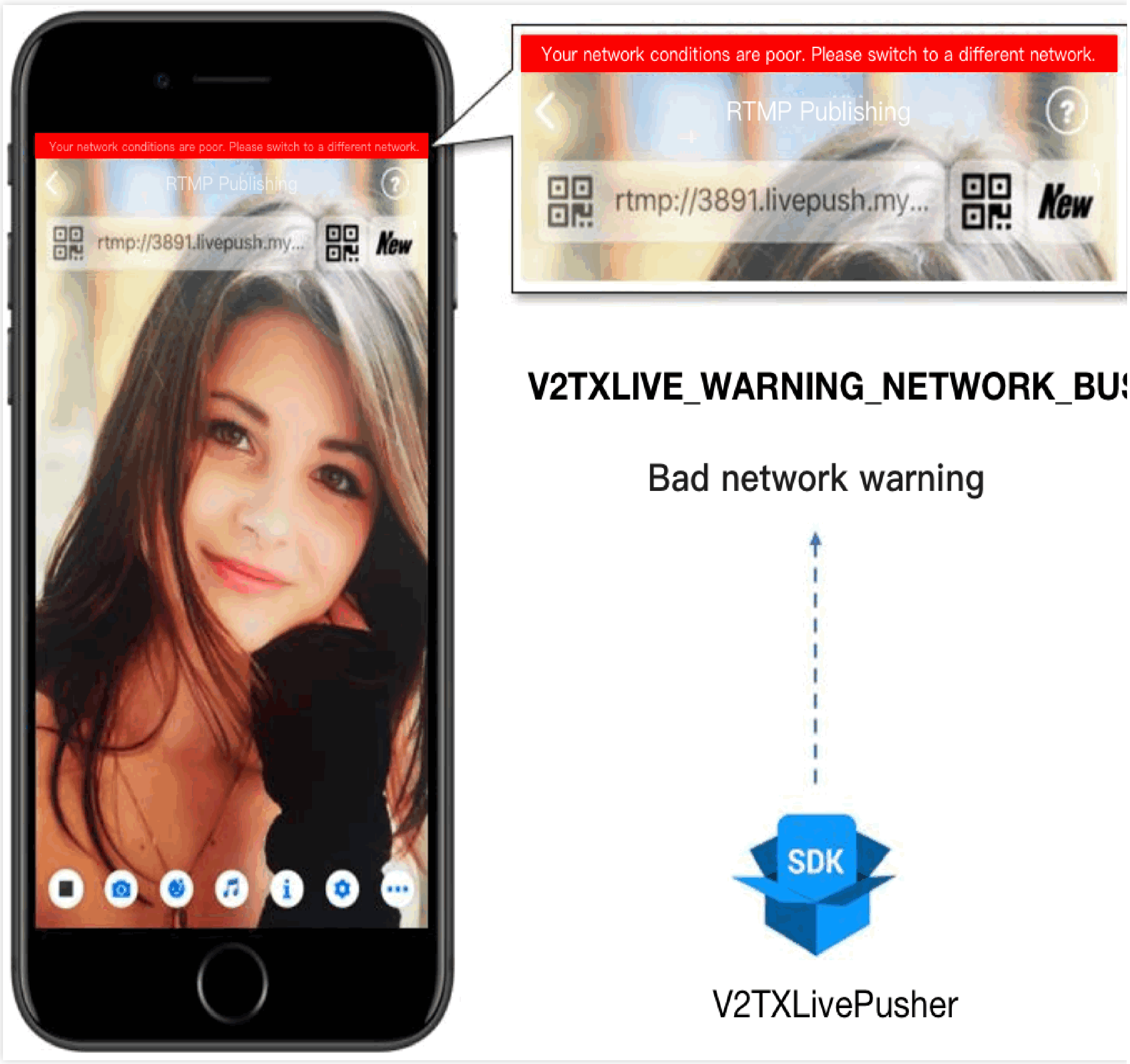
```
// Set a video watermark
mLivePusher.setWatermark(BitmapFactory.decodeResource(getResources(),R.drawable.wat
```

Step 7. Set video quality

Call `setVideoQuality` in `V2TXLivePusher` to set the quality of videos watched by audience. The encoding parameters set determine the quality of videos presented to audience. The local video watched by the host is the original HD version that has not been encoded or compressed, and is therefore not affected by the settings. For details, please see Setting Video Quality.

Step 8. Inform hosts of poor network conditions

Connecting phones to Wi-Fi does not necessarily guarantee network conditions. In case of poor Wi-Fi signal or limited bandwidth, the network speed of a Wi-Fi connected phone may be slower than that of a phone using 4G. Hosts should be informed when their network conditions are bad and be prompted to switch to a different network.


Your network conditions are poor. Please switch to a different network.
RTMP Publishing
rtmp://3891.livepush.my...
New
V2TXLIVE_WARNING_NETWORK_BU
Bad network warning
SDK
V2TXLivePusher

You can capture the **V2TXLIVE_WARNING_NETWORK_BUSY** event using onWarning in `V2TXLivePusherObserver` . The event indicates poor network conditions for hosts, which result in stuttering for audience. When this event occurs, you can send a UI message about poor network conditions to hosts, as shown above.

```
@Override
public void onWarning(int code, String msg, Bundle extraInfo) {
    if (code == V2TXLiveCode.V2TXLIVE_WARNING_NETWORK_BUSY) {
        showNetBusyTips(); // Show a “network busy” message
    }
}
```

Step 9. Set the orientation

In most cases, hosts stream while holding their phones vertically, and audience watch videos in portrait resolutions (e.g., 540 × 960). However, there are also cases where hosts hold phones horizontally, and ideally, audience should watch videos in landscape resolutions (960 × 540), as shown below:
By default, `V2TXLivePusher` outputs videos in portrait resolutions. You can output landscape-mode videos to audience by modifying a parameter of the setVideoQuality API.

```
mLivePusher.setVideoQuality(mVideoResolution, isLandscape ? V2TXLiveVideoResolution
```

Step 10. Stop publishing streams

As there can be only one `V2TXLivePusher` object running at a time, make sure that you release all the resources when stopping publishing.

```
// Stop screen sharing and release the resources
public void stopPublish() {
    mLivePusher.stopScreenCapture();
    mLivePusher.setObserver(null);
    mLivePusher.stopPush();
}
```

Event Handling

Listening for events

The SDK listens for publishing events and errors via the V2TXLivePusherObserver delegate. See V2TXLiveCode for a detailed list of events and error codes.

Errors

An error indicates that the SDK encountered a serious problem that made it impossible for stream publishing to continue.

Event ID	Code	Description
V2TXLIVE_ERROR_FAILED	-1	A common error not yet classified
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	An invalid parameter was passed in during API calling.
V2TXLIVE_ERROR_REFUSED	-3	The API call was rejected.
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	The API cannot be called.

V2TXLIVE_ERROR_INVALID_LICENSE	-5	Failed to call the API due to invalid license.
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	The server request timed out.
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	The server could not handle your request.

Warnings

A warning indicates that the SDK encountered a problem whose severity level is warning. Warning events trigger tentative protection or recovery logic and can often be resolved.

Event ID	Code	Description
V2TXLIVE_WARNING_NETWORK_BUSY	1101	Bad network connection: data upload blocked due to limited upstream bandwidth.
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	Stuttering during video playback.
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	Failed to turn the camera on.
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	The camera is occupied. Try a different camera.
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	No access to the camera. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	Failed to turn the mic on.
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	The mic is occupied. This occurs when, for example, the user is having a call on the mobile device.
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	No access to the mic. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	The system does not support screen sharing.
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	Failed to start screen recording. If this occurs on a mobile device, it may be

		because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	Screen recording was stopped by the system.

Web

Last updated：2024-01-13 15:49:41

The TXLivePusher SDK is mainly used to publish streams for LEB (ultra-low latency streaming). It can publish audio and video the browser captures from the camera, screen, or a local media file to live streaming servers via WebRTC.

Note：

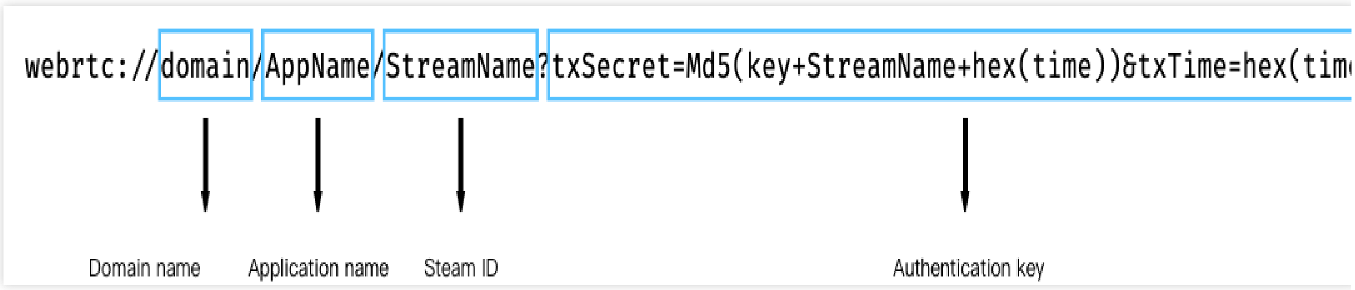
With WebRTC, each domain name can be used for the publishing of up to **concurrent 100 streams** by default. If you want to publish more streams, please submit a ticket.

Basics

Below are some basics you need to know before integrating the SDK.

Splicing publishing URLs

To use Tencent Cloud live streaming services, you need to splice publishing URLs in the format required by Tencent Cloud, which consists of four parts.


webrtc://domain/AppName/StreamName?txSecret=Md5(key+StreamName+hex(time))&txTime=hex(tim
Domain name
Application name
Steam ID
Authentication key

An authentication key is not required. You can enable publishing authentication if you need hotlink protection. For details, please see Splicing CSS URLs.

Browser support

Publishing for LEB relies on WebRTC and therefore can only be used on OS and browsers that support WebRTC.
The audio/video capturing feature is poorly supported on mobile browsers. For example, mobile browsers do not support screen recording, and only iOS 14.3 and above allow requesting camera access. Therefore, the publishing SDK is mainly used on desktop browsers. The latest version of Chrome, Firefox, and Safari all support publishing for LEB.
To publish streams on mobile browsers, use the MLVB SDK.

SDK Integration

Step 1. Prepare the page

Add an initialization script to the (desktop) page from which streams are to be published.

```
<script src="https://video.sdk.qcloudecdn.com/web/TXLivePusher-2.1.0.min.js" charse
```

Note：

The script needs to be imported into the `body` part of the HTML code. If it is imported into the `head` part, an error will be reported.

Step 2. Add a container to the HTML page

Add a player container to the section of the page where local video is to be played. This is achieved by adding a div and giving it a name, for example, `id_local_video` . Local video will be rendered in the container. To adjust the size of the container, style the div using CSS.

```
<div id="id_local_video" style="width:100%;height:500px;display:flex;align-items:ce
```

Step 3. Publish streams

1. **Generate an instance of the publishing SDK:**

Generate an instance of the global object `TXLivePusher` . All subsequent operations will be performed via the instance.

```
var livePusher = new TXLivePusher();
```

2. **Specify the local video player container:**

Specify the div for the local video player container, which is where audio and video captured by the browser will be rendered.

```
livePusher.setRenderView('id_local_video');
```

Note：

The video element generated via `setRenderView` is unmuted by default. To mute video, obtain the video element using the code below.

```
document.getElementById('id_local_video').getElementsByTagName('video')[0].muted =
```

3. **Set audio/video quality:**

Audio/video quality should be set before capturing. You can specify quality parameters if the default settings do not meet your requirements.

```
// Set video quality
livePusher.setVideoQuality('720p');
// Set audio quality
livePusher.setAudioQuality('standard');
// Set the frame rate
livePusher.setProperty('setVideoFPS', 25);
```

4. **Capture streams:**

You can capture streams from the camera, mic, screen and local media files. If capturing is successful, the player container will start playing the audio/video captured.

```
// Turn the camera on
livePusher.startCamera();
// Turn the mic on
livePusher.startMicrophone();
```

5. **Publish streams:**

Pass in the LEB publishing URL to start publishing streams. For the format of publishing URLs, please see Splicing CSS URLs. You need to replace the prefix `rtmp://` with `webrtc://` .

```
livePusher.startPush('webrtc://domain/AppName/StreamName?txSecret=xxx&txTime=xxx');
```

Note：

Before publishing, make sure that audio/video streams are captured successfully, or you will fail to call the publishing API. You can use the code below to publish streams automatically after audio/video is captured, that is, after the callback for capturing the first audio or video frame is received. If both audio and video are captured, publishing starts only after both the callback for capturing the first audio frame and that for the first video frame are received.

```
var hasVideo = false;
var hasAudio = false;
var isPush = false;
livePusher.setObserver({
onCaptureFirstAudioFrame: function() {
    hasAudio = true;
    if (hasVideo && !isPush) {
isPush = true;
livePusher.startPush('webrtc://domain/AppName/StreamName?txSecret=xxx&txTime=xxx');
    }
},
onCaptureFirstVideoFrame: function() {
    hasVideo = true;
    if (hasAudio && !isPush) {
isPush = true;
livePusher.startPush('webrtc://domain/AppName/StreamName?txSecret=xxx&txTime=xxx');
    }
}
});
```

6. **Stop publishing:**

```
livePusher.stopPush();
```

7. **Stop capturing audio and video:**

```
// Turn the camera off
livePusher.stopCamera();
// Turn the mic off
livePusher.stopMicrophone();
```

Advanced Features

Compatibility

The SDK provides a static method to check whether a browser supports WebRTC.

```
TXLivePusher.checkSupport().then(function(data) {
    // Whether WebRTC is supported
    if (data.isWebRTCSupported) {
        console.log('WebRTC Support');
    } else {
        console.log('WebRTC Not Support');
    }
    // Whether H.264 is supported
    if (data.isH264EncodeSupported) {
        console.log('H264 Encode Support');
    } else {
        console.log('H264 Encode Not Support');
    }
});
```

Event callbacks

The SDK supports callback event notifications. You can set an observer to receive callbacks of the SDK's status and WebRTC-related statistics. For details, see TXLivePusherObserver.

```
livePusher.setObserver({
    // Warnings for publishing
    onWarning: function(code, msg) {
        console.log(code, msg);
    },
    // Publishing status
    onPushStatusUpdate: function(status, msg) {
        console.log(status, msg);
    },
    // Publishing statistics
    onStatisticsUpdate: function(data) {
        console.log('video fps is ' + data.video.framesPerSecond);
    }
});
```

Device management

You can use a device management instance to get the device list, switch devices, and perform other device-related operations.

```
var deviceManager = livePusher.getDeviceManager();
// Get the device list
deviceManager.getDevicesList().then(function(data) {
    data.forEach(function(device) {
        console.log(device.deviceId, device.deviceName);
    });
});
// Switch cameras
deviceManager.switchCamera('camera_device_id');
```

Flutter

Publishing from Camera

Last updated：2024-01-13 15:49:41

Feature Overview

Publishing from the camera refers to the process of collecting video and audio data from the mobile phone's camera and mic, encoding the data, and pushing it to cloud-based live streaming platforms. Tencent Cloud live_flutter_plugin provides the camera push capabilities via the `v2_tx_live_pusher` APIs.

Notes

About running projects on x86 emulators: The SDK uses many audio and video APIs of the iOS system, most of which cannot be used on the x86 emulator built into macOS. Therefore, we recommend that you test your project on a real device.

Sample Code

Platform	GitHub Address	Key Class
iOS	GitHub	CameraPushViewController.m
Android	GitHub	CameraPushMainActivity.java
Flutter	GitHub	live_camera_push.dart

Note:
 In addition to the above sample code, regarding frequently asked questions among developers, Tencent Cloud offers an easy-to-understand API example project, which you can use to quickly learn how to use different APIs.
iOS: MLVB-API-Example
Android: MLVB-API-Example
Flutter: Live-API-Example

Getting Started

1. Set dependencies

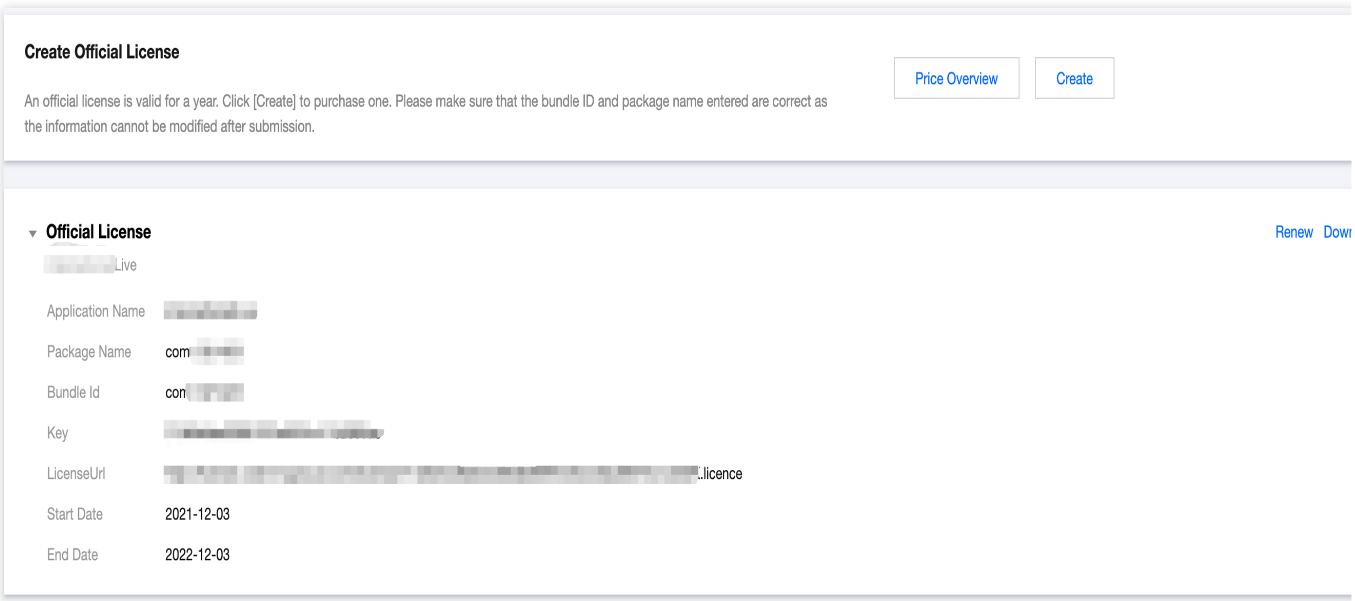
Integrate `live_flutter_plugin` into your application as instructed in SDK Integration Guide.

```
dependencies:
  live_flutter_plugin: latest version number
```

2. Configure a license for the SDK

1. Get the license:

If you have the required license, get the license URL and key in the CSS console.


Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

Live

Application Name	
Package Name	com
Bundle Id	con
Key	
LicenseUrl	.licence
Start Date	2021-12-03
End Date	2022-12-03

If you don't have the required license, apply for a license as instructed in New License and Renewal.

2. Before your application calls features of `live_flutter_plugin` , complete the following configuration:

```
import 'package:live_flutter_plugin/v2_tx_live_premier.dart';

 /// Tencent Cloud license management page (https://console.tencentcloud.com/live/l
setupLicense() {
  // The license URL of the current application
  var LICENSEURL = "";
  // The license key of the current application
  var LICENSEURLKEY = "";
  V2TXLivePremier.setLicence(LICENSEURL, LICENSEURLKEY);
}
```

Note:

The `packageName/BundleId` configured in the license must be the same as that of the application; otherwise, stream push will fail.

3. Initialize the `V2TXLivePusher` component

Create a `V2TXLivePusher` object and specify `V2TXLiveMode` .

```
import 'package:live_flutter_plugin/v2_tx_live_pusher.dart';

/// Initialize `V2TXLivePusher`
initPusher() {
  _livePusher = V2TXLivePusher(V2TXLiveMode.v2TXLiveModeRTC);
}
```

4. Set the video rendering view

```
import 'package:live_flutter_plugin/widget/v2_tx_live_video_widget.dart';

/// The video rendering view widget
Widget renderView() {
  return V2TXLiveVideoWidget(
    onViewCreated: (viewId) async {
      /// Set the video rendering view
      _livePusher.setRenderViewID(_renderViewId);
      /// Enable camera preview
      _livePusher.startCamera(true);
    },
  );
}
```

5. Start and stop publishing

After calling `startCamera` to enable camera preview, you can call the startPush API in `V2TXLivePusher` to start publishing. You can use TRTC's URL or an RTMP URL for publishing. The former uses UDP. It offers better streaming quality and supports co-anchoring.

```
/// Start stream push
startPush() async {
  // Generate a stream push address of RTMP/TRTC
  var url = "";
  // Start stream push
  await _livePusher.startPush(url);
    // Turn the mic on
  await _livePusher.startMicrophone();
}
```

After stream push ends, you can call the stopPush API in `V2TXLivePusher` to stop stream push.

```
/// Stop stream push
```

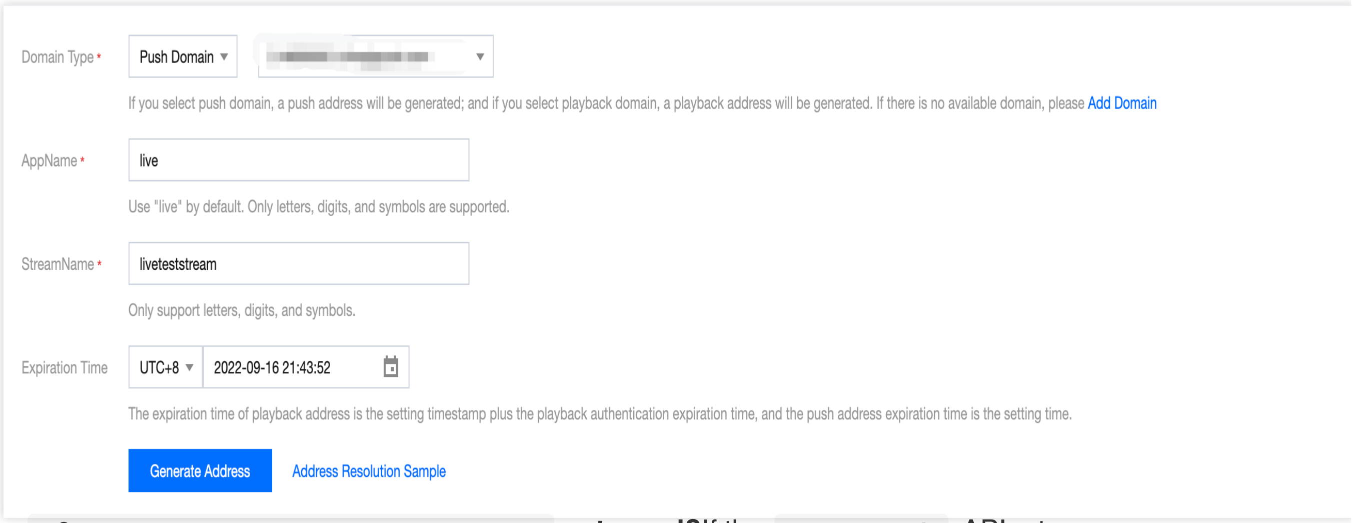
```
stopPush() async {
  // Turn the camera off
  await _livePusher.stopCamera();
  // Turn the mic off
  await _livePusher.stopMicrophone();
  // Stop stream push
  await _livePusher.stopPush();
}
```

Note:

If you have enabled camera preview, please disable it when you stop publishing streams.

How do I get a valid stream push URL?

Activate CSS. In the **CSS console**, go to **Auxiliary Tools** > **Address Generator** to generate a stream push URL. For more information, see Publishing/Playback URL.



Domain Type * Push Domain [redacted]

If you select push domain, a push address will be generated; and if you select playback domain, a playback address will be generated. If there is no available domain, please Add Domain

AppName * live

Use "live" by default. Only letters, digits, and symbols are supported.

StreamName * liveteststream

Only support letters, digits, and symbols.

Expiration Time UTC+8 2022-09-16 21:43:52

The expiration time of playback address is the setting timestamp plus the playback authentication expiration time, and the push address expiration time is the setting time.

Why is `V2TXLIVE_ERROR_INVALID_LICENSE` returned?If the `startPush` API returns `V2TXLIVE_ERROR_INVALID_LICENSE` , it means your license verification failed. Please check your configuration against Step 2. Configure a license for the SDK.

6. Publish audio-only streams

If your live streaming scenarios involve audio only, you can skip Step 4 or do not call `startCamera` before `startPush` .

```
/// Start stream push
startPush() async {
  // Initialize `V2TXLivePusher`
  _livePusher = V2TXLivePusher(V2TXLiveMode.v2TXLiveModeRTC);
  // Generate a stream push address of RTMP/TRTC
  var url = "";
  // Start stream push
  await _livePusher.startPush(url);
```

```
    // Turn the mic on
  await _livePusher.startMicrophone();
}
```

Note:

If you publish audio-only streams but no streams can be pulled from an RTMP, FLV, or HLS playback URL, there is a problem with your line configuration. Please submit a ticket for help.

7. Set video quality

You can call the setVideoQuality API in `V2TXLivePusher` to set image definition on the viewer end. The video image watched by the host is the source video without encoding or compression and is not subject to settings. However, viewers can perceive the encoding quality of the video encoder set in `setVideoQuality` . For more information, see Setting Video Quality.

8. Set the beauty filter style and skin brightening and rosy skin effects

You can call the getBeautyManager API in `V2TXLivePusher` to get the `TXBeautyManager` instance so as to further set the beauty filter effect.

Beauty filter style

The SDK has three built-in skin smoothing algorithms, each of which corresponds to a beauty filter style. You can select the one most suitable for your product needs. For more information, see the TXBeautyManager.h file.

Beauty Filter Style	Description
TXBeautyStyleSmooth	The smooth style, which features more obvious skin smoothing effects and is suitable for live showrooms.
TXBeautyStyleNature	The natural style, which retains more facial details and is more natural.
TXBeautyStylePitu	The Pitu style, which uses the beauty filter algorithm developed by YouTu Lab. Its effect combines the smooth style and the natural style, that is, it retains more skin details than the smooth style and delivers more obvious skin smoothing effects than the natural style.

You can call the setBeautyStyle API in `TXBeautyManager` to set the beauty filter style.

Item	Configuration	Description
Beauty filter strength	Via the setBeautyLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the more obvious the effect.
Skin brightening filter strength	Via the setWhitenessLevel API in \`TXBeautyManager\`	Value range: 0-9. \`0\` means the filter is disabled. The greater the value, the

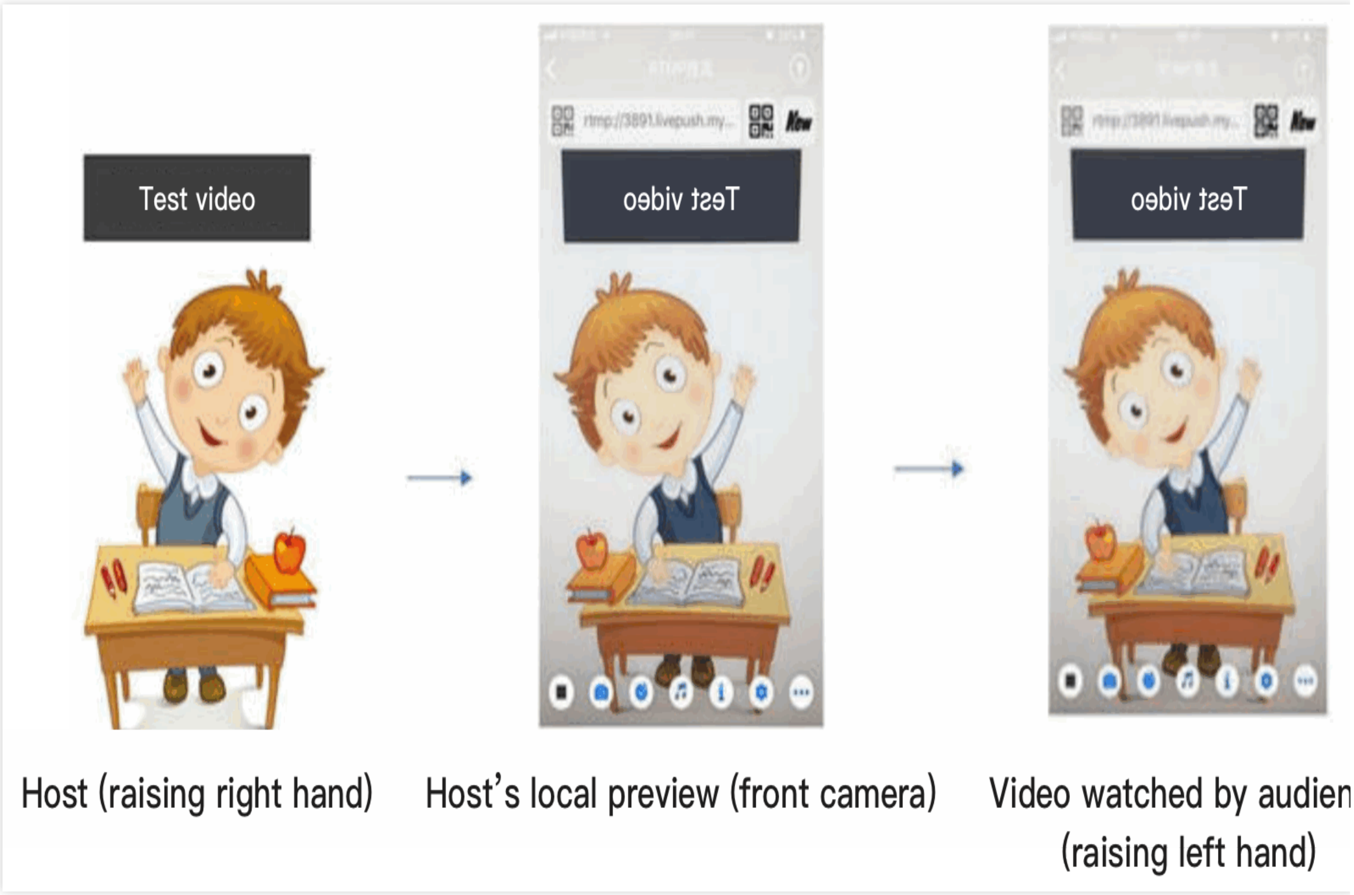
		more obvious the effect.
Rosy skin filter strength	Via the setRuddyLevel API in `TXBeautyManager`	Value range: 0-9. `0` means the filter is disabled. The greater the value, the more obvious the effect.

9. Manage devices

`V2TXLivePusher` offers a set of APIs for device control. You can use `getDeviceManager` to get the `TXDeviceManager` instance for device management. For detailed directions, see TXDeviceManager API.

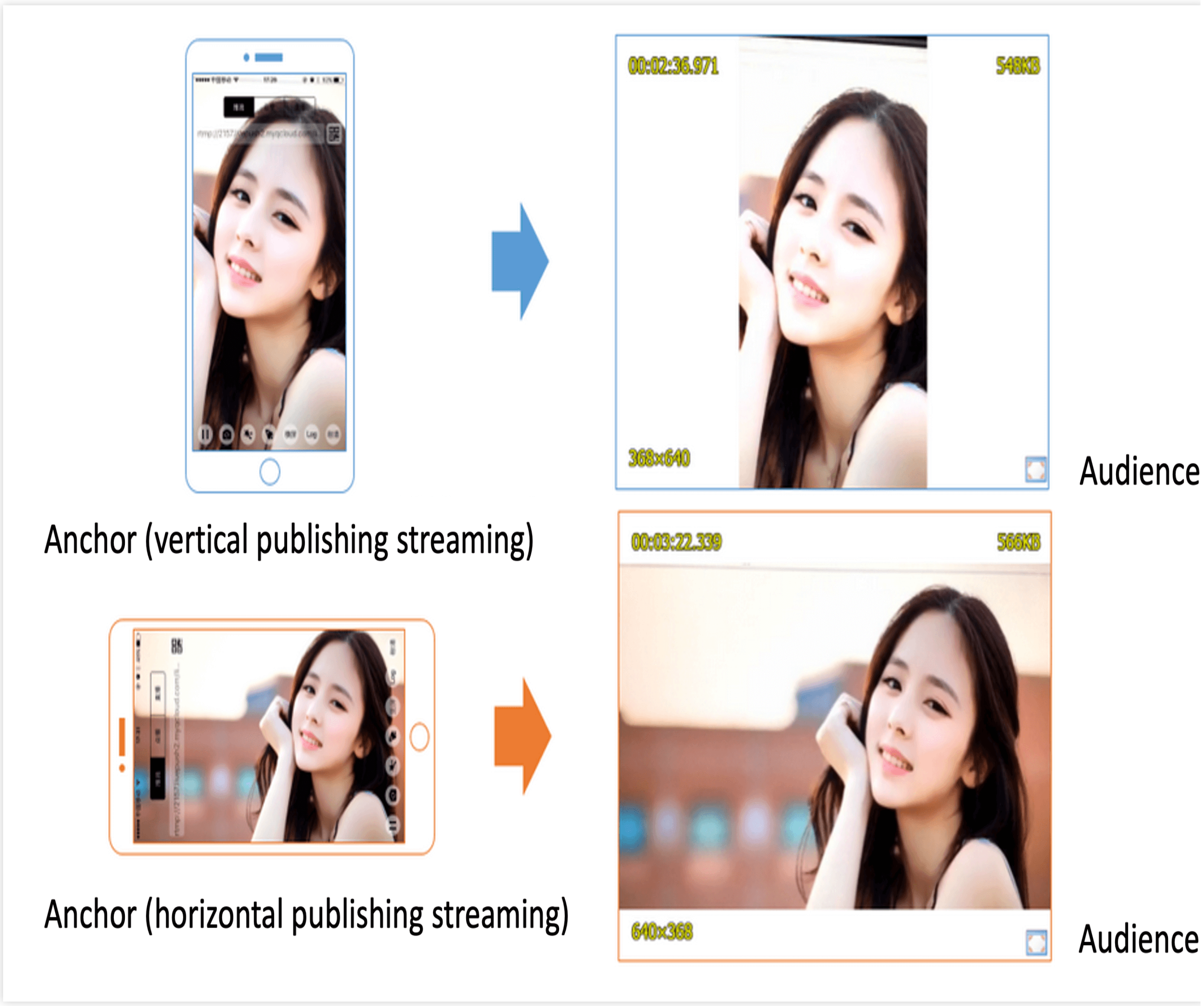
10. Set the mirror effect on the audience side

You can call setRenderMirror of `V2TXLivePusher` to change the camera mirroring mode, so as to change the mirroring effect of the video image seen by viewers. If the host uses the front camera for live streaming, the image will be reversed by the SDK by default.



11. Publish streams in landscape mode

In most cases, hosts stream while holding their phones vertically, and audience watch videos in portrait resolutions (e.g., 540 × 960). However, there are also cases where hosts hold phones horizontally, and ideally, audience should watch videos in landscape resolutions (960 × 540).


00:02:36.971
548KB
368x640
Audience
Anchor (vertical publishing streaming)
00:03:22.339
566KB
640x368
Anchor (horizontal publishing streaming)
Audience

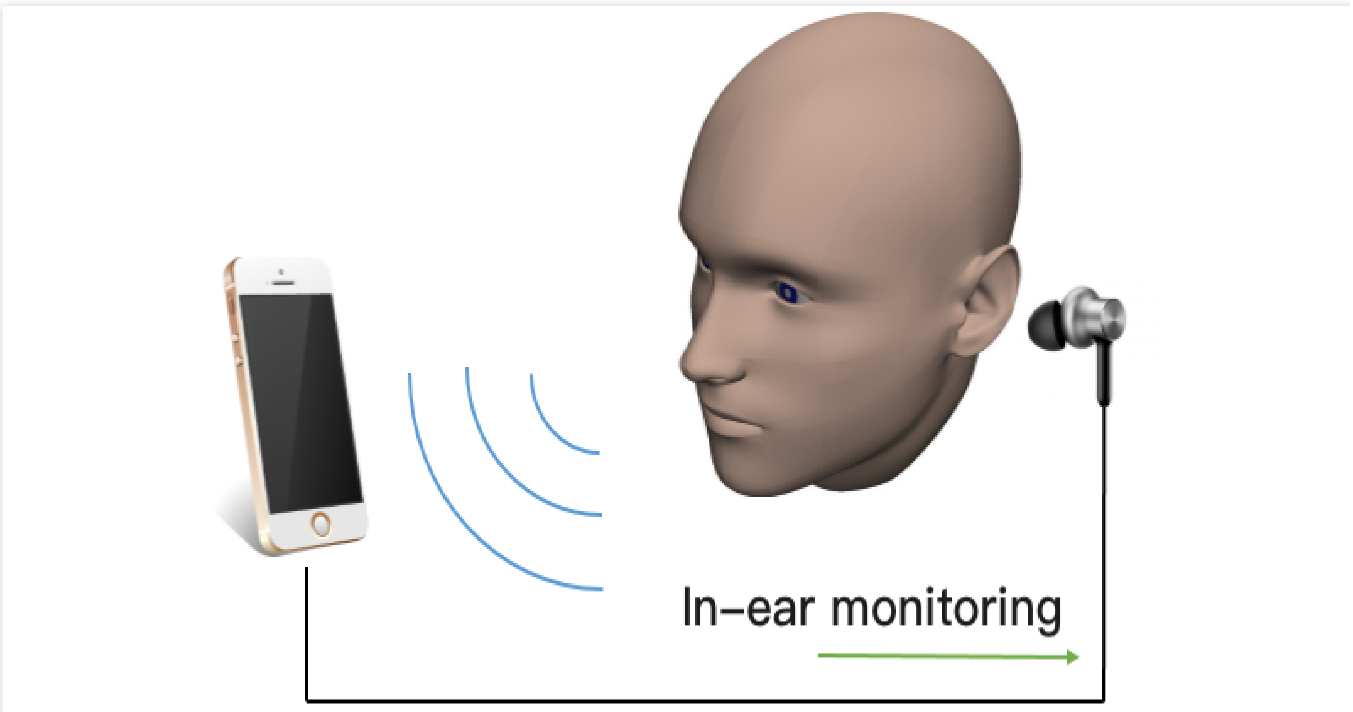
`V2TXLivePusher` pushes video images in portrait mode by default. To push video images in landscape mode, you can modify the parameters in the setVideoQuality API to set the image orientation on viewers' devices.

```
// Video encoding parameters
var param = V2TXLiveVideoEncoderParam();
param.videoResolutionMode = isLandscape ? V2TXLiveVideoResolutionMode.v2TXLiveVideo
_livePusher.setVideoQuality(param);
```

12. Set audio effects

Call getAudioEffectManager in `V2TXLivePusher` to get a `TXAudioEffectManager` instance, which can be used to mix background music and set in-ear monitoring, reverb, and other audio effects. Background music mixing

means mixing into the published stream the music played by the host's phone so that the audience can also hear the music.
Call the `enableVoiceEarMonitor` API in `TXAudioEffectManager` to enable in-ear monitoring, which allows hosts to hear their vocals in earphones when they sing.
Call the `setVoiceReverbType` API in `TXAudioEffectManager` to add reverb effects such as karaoke, hall, husky, and metal. The effects are applied to the videos watched by the audience.
Call the `setVoiceChangerType` API in `TXAudioEffectManager` to add voice changing effects such as little girl and middle-aged man to enrich host-audience interaction. The effects are applied to the videos watched by the audience.



In–ear Monitoring Illustration

Note:
For detailed directions, see TXAudioEffectManager API.

Event Handling

Listening for events

The SDK listens on push events and errors via the V2TXLivePusherObserver delegate. See v2_tx_live_code library for a detailed list of events and error codes.

Errors

An error indicates that the SDK encountered a serious problem that made it impossible for stream publishing to continue.

Event ID	Code	Description
V2TXLIVE_ERROR_FAILED	-1	A common unclassified error occurred.
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	An invalid parameter was passed in during API calling.
V2TXLIVE_ERROR_REFUSED	-3	The API call was rejected.
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	The API cannot be called.
V2TXLIVE_ERROR_INVALID_LICENSE	-5	Failed to call the API due to invalid license.
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	The server request timed out.
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	The server could not handle your request.

Warnings

A warning indicates that the SDK encountered a problem whose severity level is warning. Warning events trigger tentative protection or recovery logic and can often be resolved.

Event ID	Code	Description
V2TXLIVE_WARNING_NETWORK_BUSY	1101	Bad network connection: data upload blocked due to limited upstream bandwidth.
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	Latency during video playback.
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	Failed to turn the camera on.
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	The camera is occupied. Try a different camera.
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	No access to the camera. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	Failed to turn the mic on.
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	The mic is occupied. This occurs when, for example, the

		user is having a call on the mobile device.
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	No access to the mic. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	The system does not support screen sharing.
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	Failed to start screen recording. If this occurs on a mobile device, it may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	Screen recording was stopped by the system.

Publishing from Screen

Last updated：2024-01-13 15:49:41

Overview

Publishing from the camera refers to the process of collecting video and audio data from the mobile phone's camera and mic, encoding the data, and pushing it to cloud-based live streaming platforms. Tencent Cloud `live_flutter_plugin` provides the camera push capabilities via `V2TXLivePusher` APIs.

Sample Code

Tencent Cloud offers an easy-to-understand API example project to help you quickly learn how to use different APIs.

Platform	GitHub Address
iOS	GitHub
Android	GitHub
Flutter	GitHub

Environment Requirements

Android

Screen recording is supported in Android 5.0 and above.
Floating windows need to be enabled manually on some mobile phones and systems.

iOS

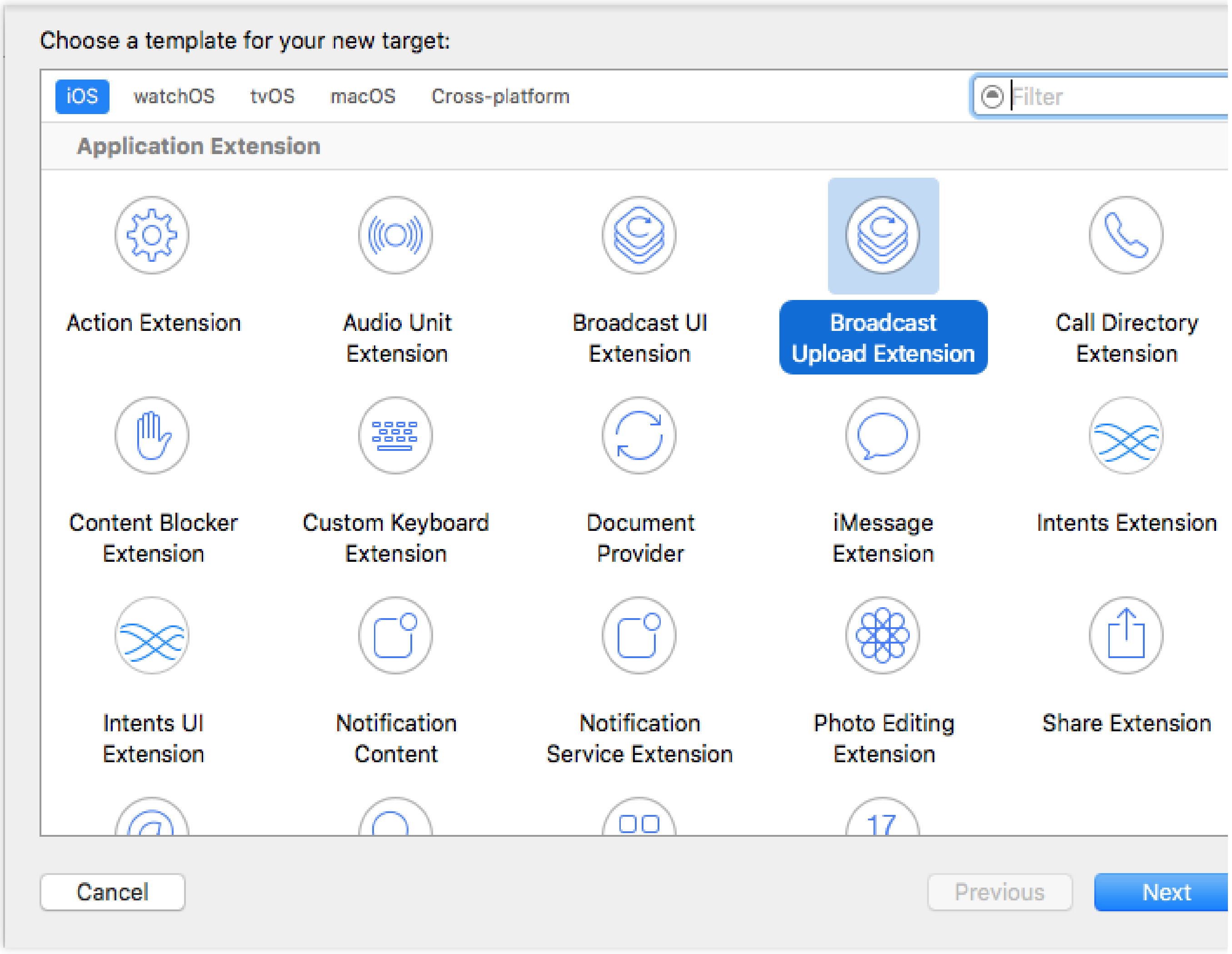
Screen recording is a new feature in iOS 10. In addition to using ReplayKit to record video from the screen, which is possible in iOS 9, with iOS 10, users can also stream live video from the screen. For details, see Go Live with ReplayKit. In iOS 11, Apple made ReplayKit more usable and more universally applicable and launched ReplayKit2, going from supporting ReplayKit alone to allowing the recording of the entire screen. Therefore, we recommend using ReplayKit2 in iOS 11 to enable the screen sharing feature. Screen sharing relies on extensions, which operate as independent processes. However, to ensure system smoothness, iOS allocates limited resources to extensions and may kill extensions with high memory usage. Given this, Tencent Cloud has further reduced the memory usage of LiteAVSDK while retaining its high streaming quality and low latency to ensure the stability of extensions.

Note:

This document describes how to use ReplayKit 2 on iOS 11 to push streams from the screen. The parts about the use of the SDK also apply to other custom stream push scenarios. For more information, see the code sample in the `Live Demo Screen` folder of the demo.

1. **Create the live streaming extension**

Open your project with Xcode and select **New** > **Target...** > **Broadcast Upload Extension**, as shown below.



Enter a product name and click **Finish**. A new directory with the product name entered will appear in your project. Under the directory, there is an automatically generated `SampleHandler` class, which is responsible for screen recording operations.

Note:

Xcode 9 or later is required, and your iPhone must be updated to iOS 11 or later. Screen recording is not supported on emulators.

2. **Import `TXLiteAVSDK_ReplayKitExt.framework`**

Import `TXLiteAVSDK_ReplayKitExt.framework` into the live streaming extension the same way you import a framework into the host application. The system libraries the SDK depends on are also the same. For more information, see iOS.

Getting Started

1. Set dependencies

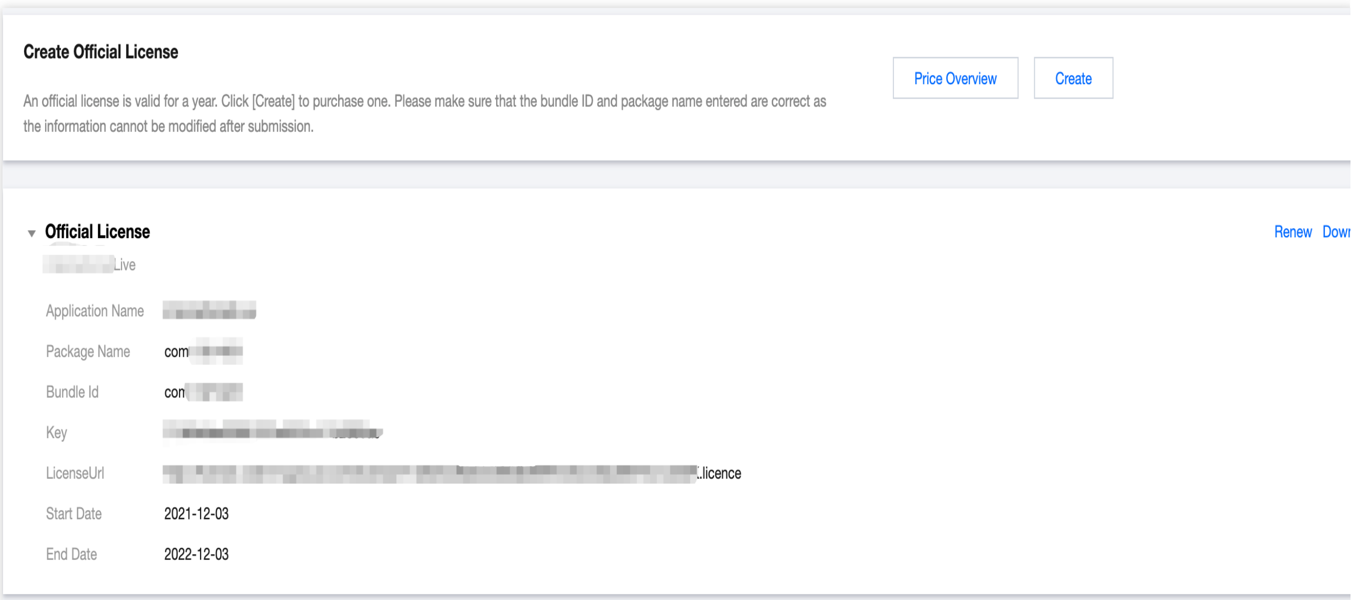
Integrate `live_flutter_plugin` into your application as instructed in SDK Integration Guide.

```
dependencies:
  live_flutter_plugin: latest version number
```

2. Configure a license for the SDK

1. Get the license:

If you have the required license, get the license URL and key in the CSS console.


Create Official License
An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.
Price Overview
Create
Official License
Live
Renew
Dowr

Application Name	
Package Name	com
Bundle Id	con
Key	
LicenseUrl	.licence
Start Date	2021-12-03
End Date	2022-12-03

If you don't have the required license, apply for a license as instructed in New License and Renewal.

2. Before your application calls features of `live_flutter_plugin`, complete the following configuration:

```
import 'package:live_flutter_plugin/v2_tx_live_premier.dart';

 /// Tencent Cloud license management page (https://console.tencentcloud.com/live/l
setupLicense() {
  // The license URL of the current application
  var LICENSEURL = "";
  // The license key of the current application
  var LICENSEURLKEY = "";
  V2TXLivePremier.setLicence(LICENSEURL, LICENSEURLKEY);
}
```

Note:

The `packageName/BundleId` configured in the license must be the same as that of the application; otherwise, stream push will fail.

3. Create a pusher object

Create a **V2TXLivePusher** object, which will be responsible for publishing operations.

```
V2TXLivePusher livePusher = V2TXLivePusher(V2TXLiveMode.v2TXLiveModeRTMP);
```

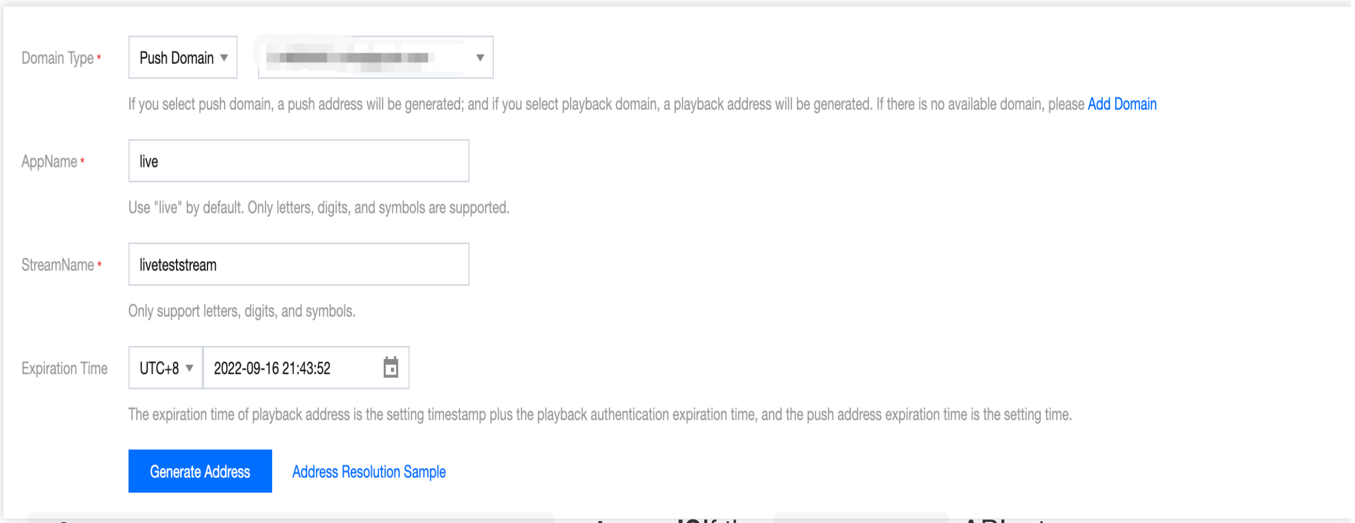
4. Start stream push

After completing Step 1, you can use the code below to start publishing streams:

```
String rtmpUrl = "rtmp://2157.livepush.myqcloud.com/live/xxxxxx";
livePusher.startMicrophone();
livePusher.startScreenCapture();
livePusher.startPush(rtmpUrl);
```

How can I obtain a valid publishing URL?

Activate CSS. In the **CSS console**, go to **Auxiliary Tools** > **Address Generator** to generate a stream push URL. For more information, see Publishing/Playback URL.



Domain Type *	Push Domain ▾ [redacted] ▾
	If you select push domain, a push address will be generated; and if you select playback domain, a playback address will be generated. If there is no available domain, please Add Domain
AppName *	live
	Use "live" by default. Only letters, digits, and symbols are supported.
StreamName *	liveteststream
	Only support letters, digits, and symbols.
Expiration Time	UTC+8 ▾ 2022-09-16 21:43:52
	The expiration time of playback address is the setting timestamp plus the playback authentication expiration time, and the push address expiration time is the setting time.

Why is `V2TXLIVE_ERROR_INVALID_LICENSE` returned?If the `startPush` API returns `V2TXLIVE_ERROR_INVALID_LICENSE`, it means your license verification failed. Please check your configuration against Step 2. Configure a license for the SDK.

5. Stop stream push

As there can be only one `V2TXLivePusher` object running at a time, make sure that you release all the resources when stopping publishing.

```
// Stop screen sharing and release the resources
void stopPush() {
    livePusher.stopMicrophone();
    livePusher.stopScreenCapture();
    livePusher.stopPush();
}
```

Event Handling

Listening for events

The SDK listens on push events and errors via the V2TXLivePusherObserver delegate. See v2_tx_live_code library for a detailed list of events and error codes.

Errors

An error indicates that the SDK encountered a serious problem that made it impossible for stream publishing to continue.

Event ID	Code	Description
V2TXLIVE_ERROR_FAILED	-1	A common unclassified error occurred.
V2TXLIVE_ERROR_INVALID_PARAMETER	-2	An invalid parameter was passed in during API calling.
V2TXLIVE_ERROR_REFUSED	-3	The API call was rejected.
V2TXLIVE_ERROR_NOT_SUPPORTED	-4	The API cannot be called.
V2TXLIVE_ERROR_INVALID_LICENSE	-5	Failed to call the API due to invalid license.
V2TXLIVE_ERROR_REQUEST_TIMEOUT	-6	The server request timed out.
V2TXLIVE_ERROR_SERVER_PROCESS_FAILED	-7	The server could not handle your request.

Warnings

A warning indicates that the SDK encountered a problem whose severity level is warning. Warning events trigger tentative protection or recovery logic and can often be resolved.

Event ID	Code	Description

V2TXLIVE_WARNING_NETWORK_BUSY	1101	Bad network connection: data upload blocked due to limited upstream bandwidth.
V2TXLIVE_WARNING_VIDEO_BLOCK	2105	Stuttering during video playback.
V2TXLIVE_WARNING_CAMERA_START_FAILED	-1301	Failed to turn the camera on.
V2TXLIVE_WARNING_CAMERA_OCCUPIED	-1316	The camera is occupied. Try a different camera.
V2TXLIVE_WARNING_CAMERA_NO_PERMISSION	-1314	No access to the camera. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_MICROPHONE_START_FAILED	-1302	Failed to turn the mic on.
V2TXLIVE_WARNING_MICROPHONE_OCCUPIED	-1319	The mic is occupied. This occurs when, for example, the user is having a call on the mobile device.
V2TXLIVE_WARNING_MICROPHONE_NO_PERMISSION	-1317	No access to the mic. This usually occurs on mobile devices and may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_NOT_SUPPORTED	-1309	The system does not support screen sharing.
V2TXLIVE_WARNING_SCREEN_CAPTURE_START_FAILED	-1308	Failed to start screen recording. If this occurs on a mobile device, it may be because the user denied the access.
V2TXLIVE_WARNING_SCREEN_CAPTURE_INTERRUPTED	-7001	Screen recording was stopped by the system.

FAQs

ReplayKit2 is a new framework introduced by Apple in iOS 11, for which relatively few official documents have been released. The framework is still being improved, and problems have been found. See below for some common questions you may have when using ReplayKit2.

1. **When does screen recording stop automatically?**

Screen recording stops automatically when the screen locks or there is an incoming call. At such times, the `broadcastFinished` function in `SampleHandler` will be invoked, and you can send a notification to users about the interruption.

2. **Why does screen recording stop sometimes during screen sharing?**

The problem usually occurs after landscape/portrait mode switch if the resolution for stream publishing is set high. The broadcast upload extension is allocated a memory of only 50 MB and will be killed if its memory usage exceeds the limit. Given this, we recommend that you set the resolution to 720p or lower.

3. **Why are images streamed from the screen of iPhone X distorted?**

iPhone X has a notch at the top of the screen, so video captured from the screen is not in the aspect ratio of 16:9. If you set the output resolution for stream publishing to 16:9, for example, to HD (960 × 540), the images published will be slightly distorted because their original aspect ratio is not 16:9. We recommend that you set the resolution according to your screen size. Besides, if you play video streamed from the screen of iPhone X in aspect fit mode, the video may have black bars, and if you play it in aspect fill mode, the video may be cropped.

Playback
iOS
LVB

Last updated：2024-01-13 15:49:41

Basics

This document introduces the live playback feature of the Video Cloud SDK.

Live streaming and video on demand

In **live streaming**, the video streams published by hosts in real time are the source of streaming. When hosts stop publishing streams, the video played stops. Since video is streamed in real time, players do not have progress bars when they play live streaming URLs.
In **video on demand (VOD)**, video files in the cloud are the source of streaming. Videos can be played at any time as long as they are not deleted from the cloud, and the playback progress can be adjusted using the progress bar. Video streaming websites such as Tencent Video and Youku Tudou are typical applications of VOD.

Supported protocols

The table below lists the common protocols used for live streaming. We recommend FLV URLs (which start with `http` and end with `flv`) for LVB and WebRTC for LEB. For more information, please see LEB.

Protocol	Pro	Con	Playback Latency
HLS	Mature, well adapted to high-concurrency scenarios	SDK integration is required.	3s - 5s
FLV	Mature, well adapted to high-concurrency scenarios	SDK integration is required	2s - 3s
RTMP	Relatively low latency	Poor performance in high-concurrency scenarios	1s - 3s
WebRTC	Lowest latency	SDK integration is required	< 1s

Note:
LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Notes

The Video Cloud SDK **does not impose any limit on the sources of playback URLs**, which means you can use it to play both Tencent Cloud and non-Tencent Cloud URLs. However, the player of the SDK supports only live streaming URLs in FLV, RTMP, HLS (M3U8), and WebRTC formats and VOD URLs in MP4, HLS (M3U8), and FLV formats.

Sample Code

Regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

Platform	GitHub Address
iOS	Github
Android	Github
Flutter	Github

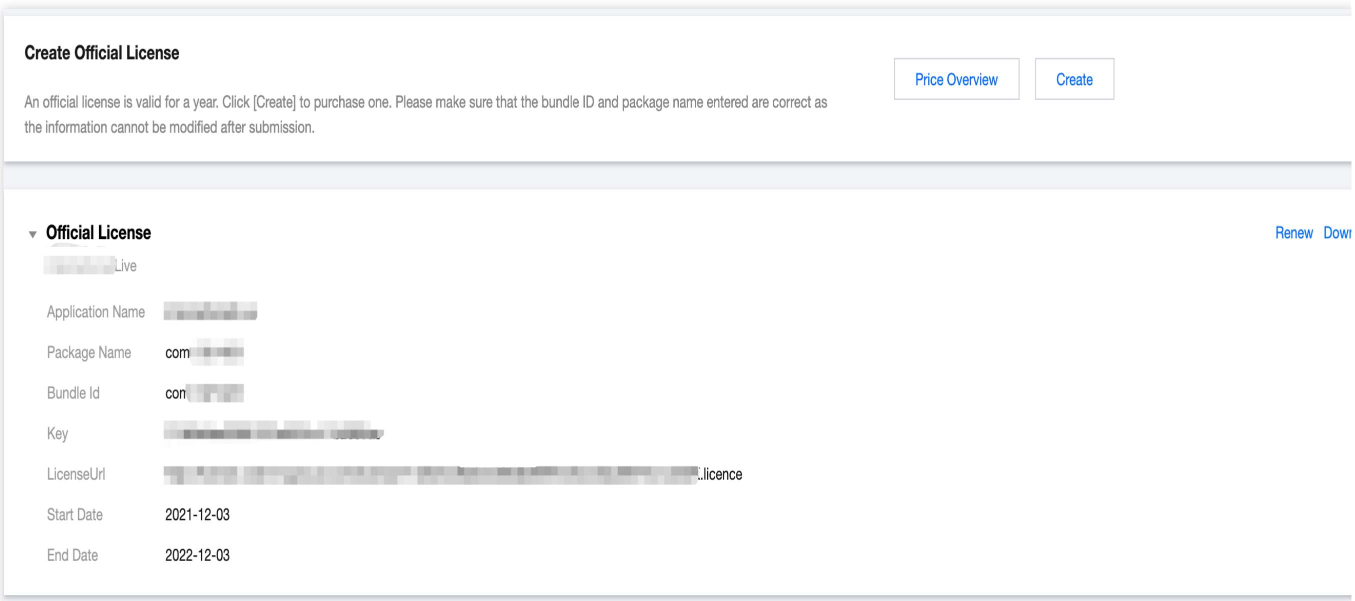
Integration

Step 1. Download the SDK

Download the SDK and follow the instructions in SDK Integration to integrate the SDK into your application.

Step 2. Configure License Authorization for SDK

1. Obtain license authorization：
If you have obtained the relevant license authorization， need to Get License URL and License Key in Cloud Live Console



Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

[redacted]Live

Application Name	[redacted]
Package Name	com[redacted]
Bundle Id	con[redacted]
Key	[redacted]
LicenseUrl	[redacted].licence
Start Date	2021-12-03
End Date	2022-12-03

If you have not yet obtained the license authorization, Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<your licenseUrl>";
    NSString * const licenceKey = @"<your key>";

    [V2TXLivePremier setEnvironment:@"GDPR"];// set your environment
    // V2TXLivePremier located in "V2TXLivePremier.h"
    [V2TXLivePremier setLicence:licenceURL key:licenceKey];
    [V2TXLivePremier setObserver:self];
    NSLog(@"SDK Version = %@", [V2TXLivePremier getSDKVersionStr]);
    return YES;
}

#pragma mark - V2TXLivePremierObserver
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
}
@end
```

Note:

The packageName configured in the license must be the same as the application itself, otherwise the play stream will fail.

Step 3. Create Player

The V2TXLivePlayer module in Tencent Cloud SDK is responsible for implementing the live broadcast function.

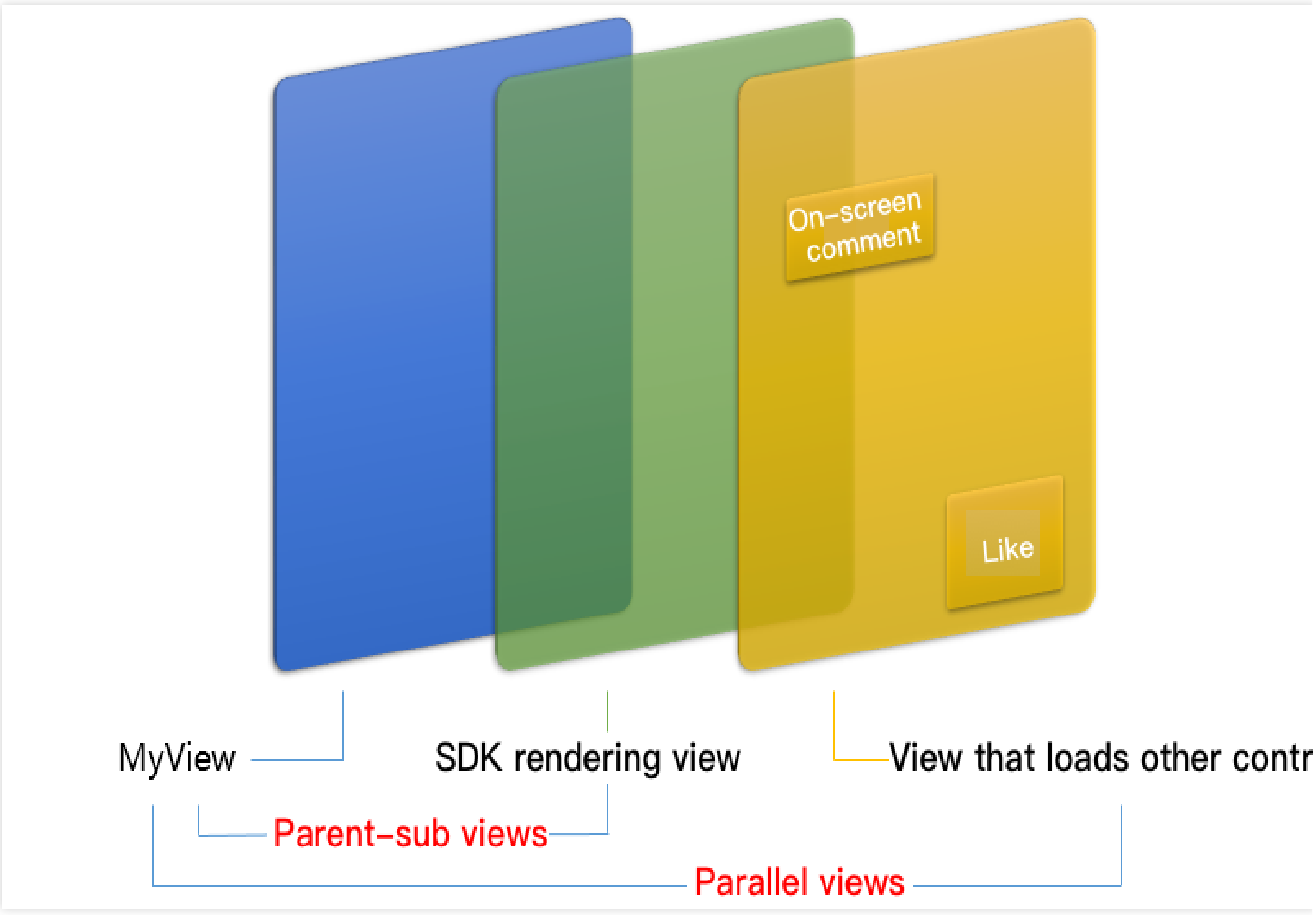
```
V2TXLivePlayer *_txLivePlayer = [[V2TXLivePlayer alloc] init];
```

Step 4. Create a rendering view

In iOS, a view is used as the basic UI rendering unit. Therefore, you need to configure a view, whose size and position you can adjust, for the player to display video images on.

```
// Use setRenderView to bind a rendering view to the player
[_txLivePlayer setRenderView:_myView];
```

Technically, the player does not render video images directly on the view (`_myView` in the sample code) you provide. Instead, it creates a subview for OpenGL rendering over the view.
You can adjust the size of video images by changing the size and position of the view. The SDK will make changes to the video images accordingly.



How can I animate views?

You are allowed great flexibility in view animation, but note that you need to modify the `transform` rather than `frame` attribute of the view.

```
[UIView animateWithDuration:0.5 animations:^{
        _myView.transform = CGAffineTransformMakeScale(0.3, 0.3); // Shrink by 1/3
}];
```

Step 5. Start playback

```
NSString* url = @"http://2157.liveplay.myqcloud.com/live/2157_xxxx.flv";
[_txLivePlayer startPlay:url];
```

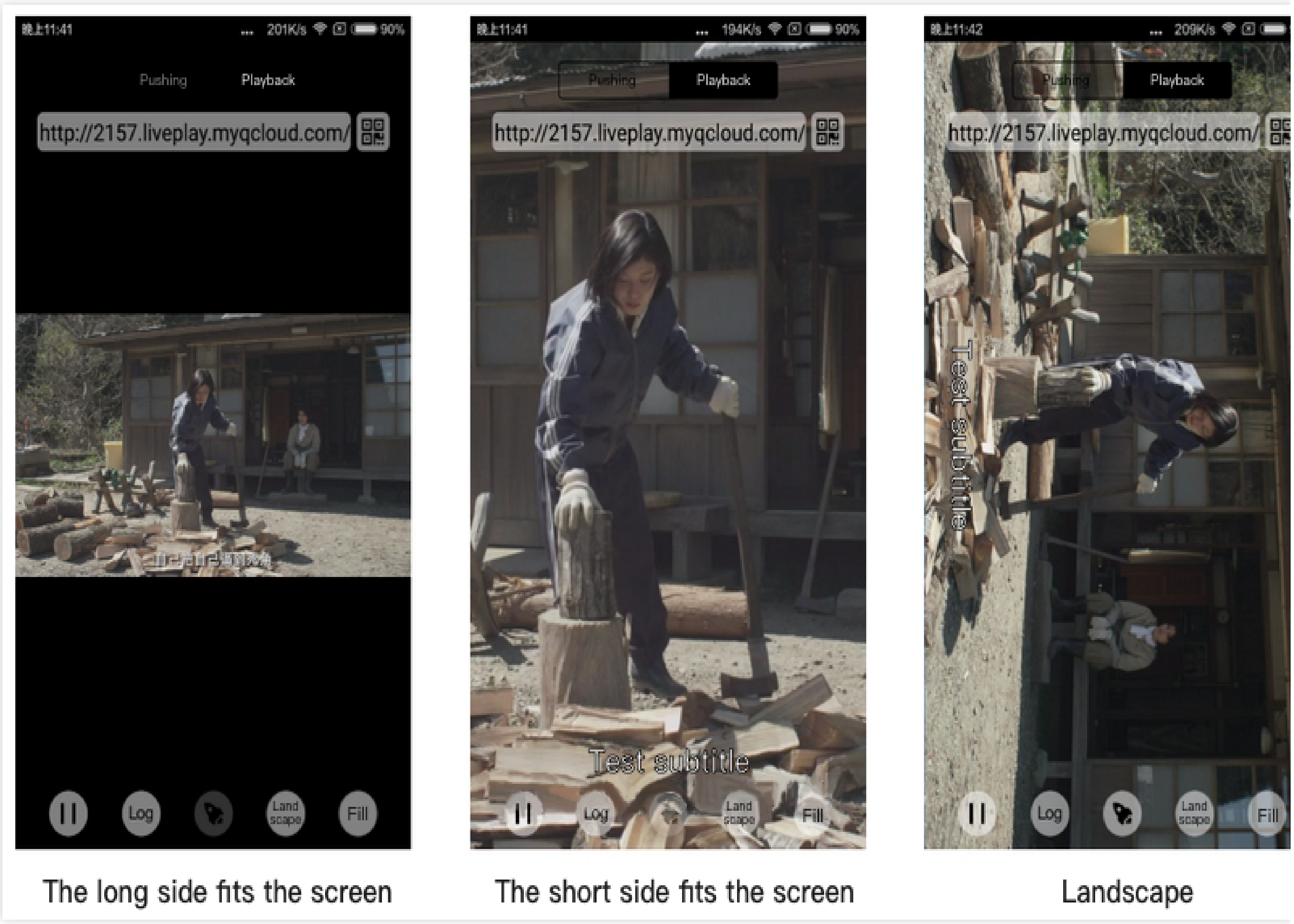
Step 6. Change the fill mode

setRenderFillMode: aspect fill or aspect fit

Value	Description
V2TXLiveFillModeFill	Images are scaled to fill the entire screen, and the excess parts are cropped. There are no black bars in this mode, but images may not be displayed in whole.
V2TXLiveFillModeFit	Images are scaled as large as the longer side can go. Neither side exceeds the screen after scaling. Images are centered, and there may be black bars.

setRenderRotation: clockwise rotation of video

Value	Description
V2TXLiveRotation0	Original
V2TXLiveRotation90	Rotate 90 degrees clockwise
V2TXLiveRotation180	Rotate 180 degrees clockwise
V2TXLiveRotation270	Rotate 270 degrees clockwise


Pushing
Playback
http://2157.liveplay.myqcloud.com/
Test subtitle
Log
Land scape
Fill
The long side fits the screen
The short side fits the screen
Landscape

Step 7. Pause playback

Technically speaking, you cannot pause a live playback. In this document, by pausing playback, we mean **freezing video** and **disabling audio**. In the meantime, new video streams continue to be sent to the cloud. When you resume playback, it starts from the time of resumption. This is in contrast to VOD. With VOD, when you pause and resume playback, the player behaves the same way as it does when you pause and resume a local video file.

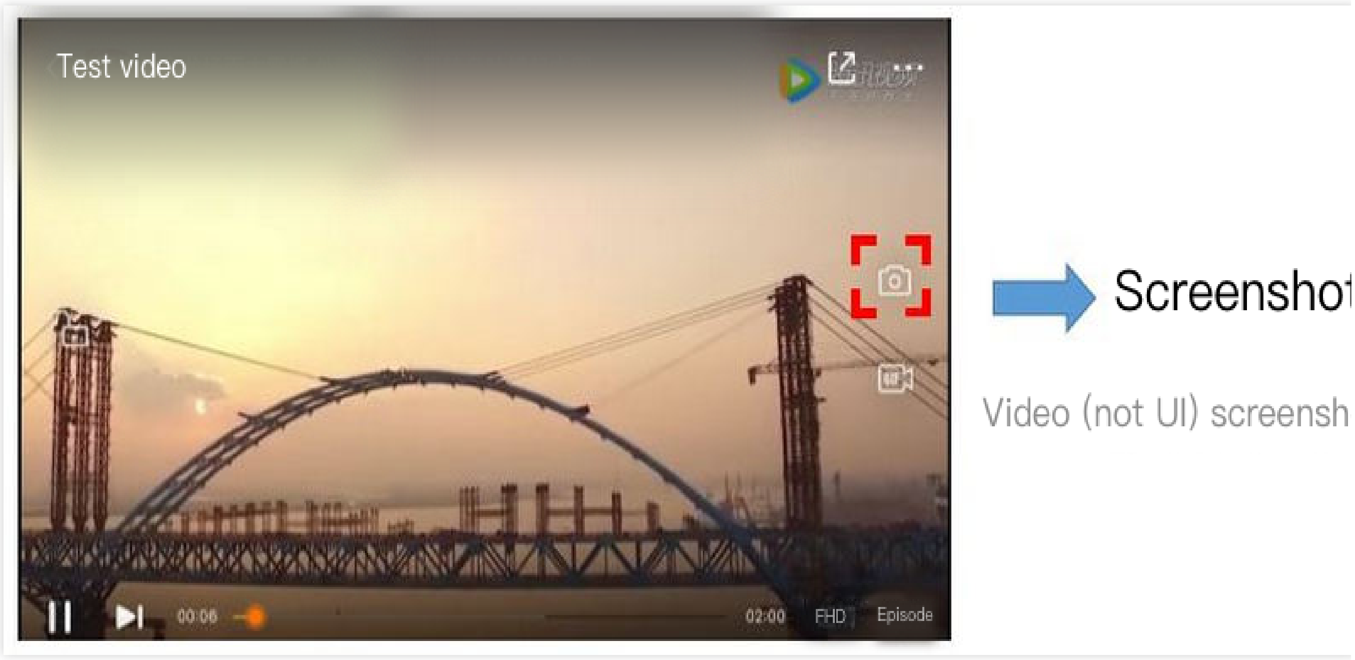
```
// Pause playback
[_txLivePlayer pauseAudio];
[_txLivePlayer pauseVideo];
// Resume playback
[_txLivePlayer resumeAudio];
[_txLivePlayer resumeVideo];
```

Step 8. Stop playback

```
// Stop playback
[_txLivePlayer stopPlay];
```

Step 9. Take a screenshot

Call **snapshot** to take a screenshot of the live video streamed. You can get the screenshot taken in the onSnapshotComplete callback of `V2TXLivePlayerObserver` . This method captures a frame of the streamed video. To capture the UI, use the corresponding API of the iOS system.



```
...
[_txLivePlayer setObserver:self];
[_txLivePlayer snapshot];
...

- (void)onSnapshotComplete:(id<V2TXLivePlayer>)player image:(TXImage *)image {
    if (image != nil) {
        dispatch_async(dispatch_get_main_queue(), ^{
            [self handle:image];
        });
    }
}
```

Latency Control

The live playback feature of the SDK is not based on FFmpeg, but Tencent Cloud’s proprietary playback engine, which is why the SDK offers better latency control than open-source players do. We provide three latency control modes, which can be used for showrooms, game streaming, and hybrid scenarios.

Comparison of the three modes

Mode	Stutter	Average Latency	Scenario	Remarks
Speedy	More likely than the speedy mode	2-3s	Live showroom (Chongding Dahui)	The mode delivers low latency and is suitable for latency-sensitive scenarios.
Smooth	Least likely of the three	>= 5s	Game streaming (Penguin Esports)	Playback is least likely to stutter in this mode, which makes it suitable for ultra-high-bitrate streaming of games such as PUBG.
Auto	Self-adaptive to network conditions	2-8s	Hybrid	The better network conditions at the audience end, the lower the latency.

Code to integrate the three modes

```
// Auto mode
[_txLivePlayer setCacheParams:1 maxTime:5];
// Speedy mode
[_txLivePlayer setCacheParams:1 maxTime:1];
// Smooth mode
[_txLivePlayer setCacheParams:5 maxTime:5];

// Start playback after configuration
```

Note:

For more information on stuttering and latency control, see Video Stutter.

Listening for SDK Events

You can bind a V2TXLivePlayerObserver to your `V2TXLivePlayer` object to receive callback notifications about the player status, playback volume, first audio/video frame, statistics, warning and error messages, etc.

Periodically triggered notifications

The onStatisticsUpdate callback notification is triggered every 2 seconds to update you on the player's status in real time. Like a car's dashboard, the callback gives you information about the SDK, such as network conditions and video information.

Parameter	Description
appCpu	CPU usage (%) of the application

systemCpu	CPU usage (%) of the system
width	Video width
height	Video height
fps	Frame rate (fps)
audioBitrate	Audio bitrate (Kbps)
videoBitrate	Video bitrate (Kbps)

The onPlayoutVolumeUpdate callback, which notifies you of the player’s volume, works only after you call enableVolumeEvaluation to enable the volume reminder. You can set the interval of the callback by specifying the `intervalMs` parameter of `enableVolumeEvaluation` .

Event-triggered notifications

Other callbacks are triggered when specific events occur.

LEB

Last updated：2024-01-13 15:49:41

LEB Overview

Live Event Broadcasting (LEB) is the ultra-low-latency version of LVB. It features lower latency than traditional streaming protocols and delivers superior playback experience with millisecond latency. It is suitable for scenarios with high requirements on latency, such as online education, sports streaming, and online quizzes.

Note:

The figure above (made using scrcpy) is a comparison of LEB and standard CDN live streaming. The images on the left and in the middle show the playback end of standard CDN live streaming and **LEB**, and the image on the right shows the publishing end.

LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Strengths

Strength	Description
Playback with millisecond latency	The latency is kept within 1s thanks to the use of UDP, as opposed to 3-5s in traditional live streaming. This, along with excellent instant streaming performance and low stuttering rate, guarantees a superior streaming experience.
Diverse features and smooth migration	LEB integrates a wide range of features including stream publishing, transcoding, recording, screenshot, porn detection, and playback. It allows smooth migration from standard live streaming.
Easy-to-use, secure, and reliable	The use of a standard protocol makes integration easy. You can play live video on Chrome and Safari without installing any plugins. In addition, the playback protocol encrypts video by default for improved security and reliability.

Use cases

Scenario	Description
Sports event	LEB offers ultra-low-latency streaming for sports events. It brings sports content to audience at low latency, allowing audience to learn what’s happening in real time.
E-commerce streaming	Some e-commerce streaming scenarios, for example, online auctions and

	sales promotion, require extremely low latency. LEB’s ability to stream at ultra-low latency ensures that hosts and audience get real-time feedback from each other, improving online shopping experience.
Online classes	LEB can be used for online classes. Its ability to stream at ultra-low latency allows teachers and students to interact with each other as they do in offline classes.
Online quizzes	Due to latency, some online quizzes have to insert extra frames at the audience end to ensure that the host and audience are in sync with each other. This is not necessary if you use LEB, whose ultra-low-latency streaming capability makes sure that the two sides are in sync. It helps you implement online quizzes more easily and deliver smoother experience.
Showrooms	LEB can significantly improve the experience of latency-sensitive interactions such as gift giving in live showrooms.

Tryout

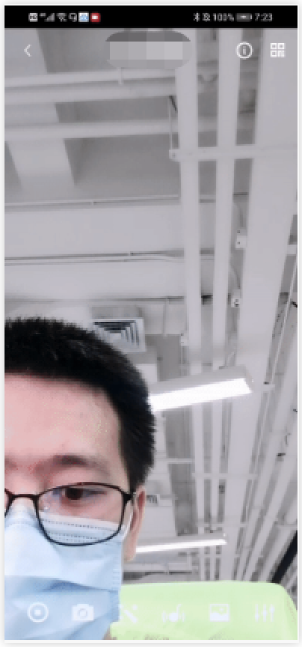
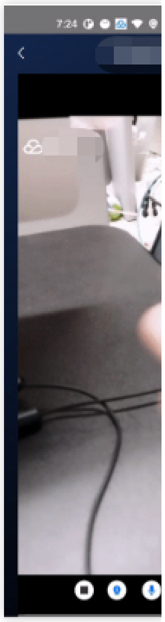
Video Cloud Toolkit is a comprehensive audio-video solution developed by Tencent Cloud that allows you to try out the features of the TRTC, MLVB and UGC SDKs, including the **LEB Player**.

Note:

The demonstration and directions in this document use the demo app for Android as an example. The UI of the app for iOS is slightly different.

Source code and demonstration

Source Code	Demo	Publishing Demonstration (Android)	Playback Demonstr
Android			
iOS			

Note:

In addition to the above sample code, regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

iOS：MLVB-API-Example

Android：MLVB-API-Example

Flutter：Live-API-Example

Publishing

LEB is compatible with LVB, which means you can publish streams using an ordinary publisher and play the streams using LEB.

1. Download and install Video Cloud Toolkit, log in, and go to **MLVB > Push (Camera)**.
2. Allow the permissions asked, and tap **Auto-generate** to start publishing streams.
3. If publishing is successful, tap the QR code icon in the top right and select **LEB** to get the playback URL for LEB.
4. During publishing, you can tap the menu icon in the bottom right to apply filters, add background music, switch cameras, etc.

Playback

1. Download and install Video Cloud Toolkit, log in, and go to **Live Broadcast > LEB Player**.
2. Allow the permissions asked, tap the scan button, and scan the LEB playback URL obtained earlier.
3. Playback starts automatically once the QR code is read. During playback, you can tap the menu icon in the bottom right to mute playback or change other settings.

Integration

In the new version of the MLVB SDK, you can use V2TXLivePlayer for LEB and `V2TXLivePusher` for publishing. LEB supports WebRTC protocols and uses the standard extension method. All URLs in LEB start with `webrtc://` .

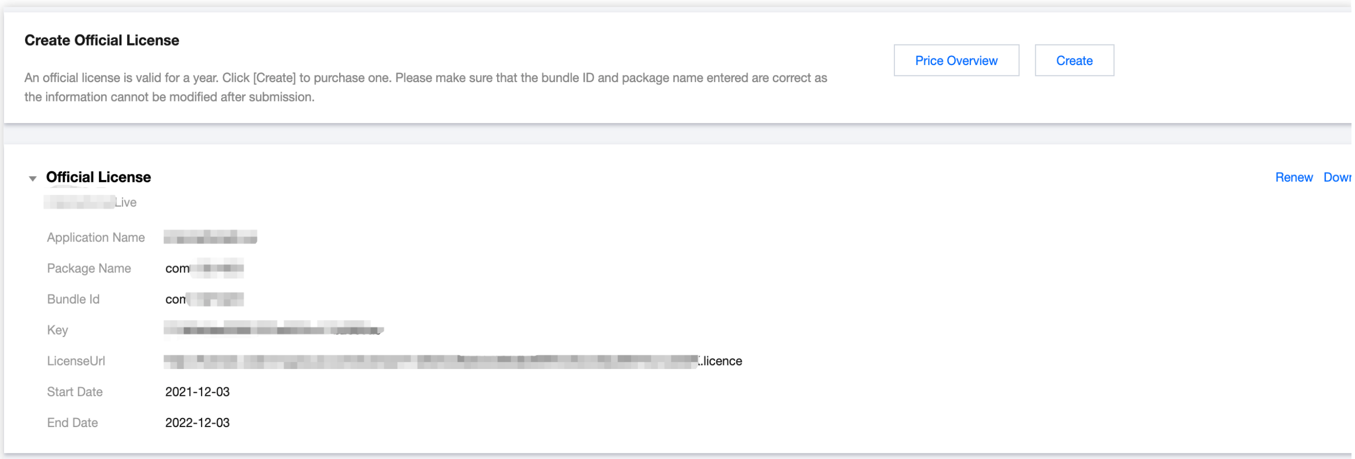
Step 1. Download the SDK

Download LiteAV_All or Professional at SDK Download.

Step 2. Configure License Authorization for SDK

1. Obtain license authorization：

If you have obtained the relevant license authorization， Need to Get License URL and License Key in Cloud Live Console



Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

Live

Application Name	
Package Name	com
Bundle Id	con
Key	
LicenseUrl	.licence
Start Date	2021-12-03
End Date	2022-12-03

If you have not yet obtained the license authorization， Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    NSString * const licenceURL = @"<your licenseUrl>";
    NSString * const licenceKey = @"<your key>";

    // V2TXLivePremier located in "V2TXLivePremier.h"
    [V2TXLivePremier setEnvironment:@"GDPR"];// set your environment
    [V2TXLivePremier setLicence:licenceURL key:licenceKey];
    [V2TXLivePremier setObserver:self];
    NSLog(@"SDK Version = %@", [V2TXLivePremier getSDKVersionStr]);
    return YES;
}

#pragma mark - V2TXLivePremierObserver
```

```
- (void)onLicenceLoaded:(int)result Reason:(NSString *)reason {
    NSLog(@"onLicenceLoaded: result:%d reason:%@", result, reason);
}
@end
```

Step 3. Get a playback URL

In live streaming, URLs are needed for both publishing and playback. For instructions on how to get URLs for LEB, please see Live Event Broadcasting (LEB) > Get Playback URL.

All LEB URLs start with `webrtc://` , as in:

```
webrtc://{Domain}/{AppName}/{StreamName}
```

The table below lists the key fields in an LEB URL and their meanings.

Field	Description
webrtc://	Prefix
Domain	Domain name
AppName	Application name, which is `live` by default. It specifies the storage path of a live streaming file.
StreamName	Stream name, which is the unique identifier of a stream

Note:

To publish streams, please see Publishing from Camera or Publishing from Screen.

Step 4. Start LEB

You can use a `V2TXLivePlayer` object for LEB. For details, see the code below (make sure that you pass in the correct URL).

Sample code

Android

iOS

```
// Create a V2TXLivePlayer object
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
player.setObserver(new MyPlayerObserver(playerView));
player.setRenderView(mSurfaceView);
// Pass in the low-latency playback URL to start playback
player.startPlay("webrtc://{Domain}/{AppName}/{StreamName}");
```

```
V2TXLivePlayer *player = [[V2TXLivePlayer alloc] init];
[player setObserver:self];
[player setRenderView:videoView];
[player startPlay:@"webrtc://{Domain}/{AppName}/{StreamName}"];
```

Android
LVB

Last updated：2024-11-18 20:26:09

Basics

This document introduces the live playback feature of the Video Cloud SDK.

Live streaming and video on demand

In **live streaming**, the video streams published by hosts in real time are the source of streaming. When hosts stop publishing streams, the video played stops. Since video is streamed in real time, players do not have progress bars when they play live streaming URLs.
In **video on demand (VOD)**, video files in the cloud are the source of streaming. Videos can be played at any time as long as they are not deleted from the cloud, and the playback progress can be adjusted using the progress bar. Video streaming websites such as Tencent Video and Youku Tudou are typical applications of VOD.

Supported protocols

The table below lists the common protocols used for live streaming. We recommend FLV URLs (which start with `http` and end with `flv`) for LVB and WebRTC for LEB. For more information, please see LEB.

Protocol	Pro	Con	Playback Latency
HLS	Mature, well adapted to high-concurrency scenarios	SDK integration is required.	3s - 5s
FLV	Mature, well adapted to high-concurrency scenarios	SDK integration is required	2s - 3s
RTMP	Relatively low latency	Poor performance in high-concurrency scenarios	1s - 3s
WebRTC	Lowest latency	SDK integration is required	< 1s

Note:
LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Notes

The Video Cloud SDK **does not impose any limit on the sources of playback URLs**, which means you can use it to play both Tencent Cloud and non-Tencent Cloud URLs. However, the player of the SDK supports only live streaming URLs in FLV, RTMP, HLS (M3U8), and WebRTC formats and VOD URLs in MP4, HLS (M3U8), and FLV formats.

Sample Code

Regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

Platform	GitHub Address
iOS	Github
Android	Github
Flutter	Github

Integration

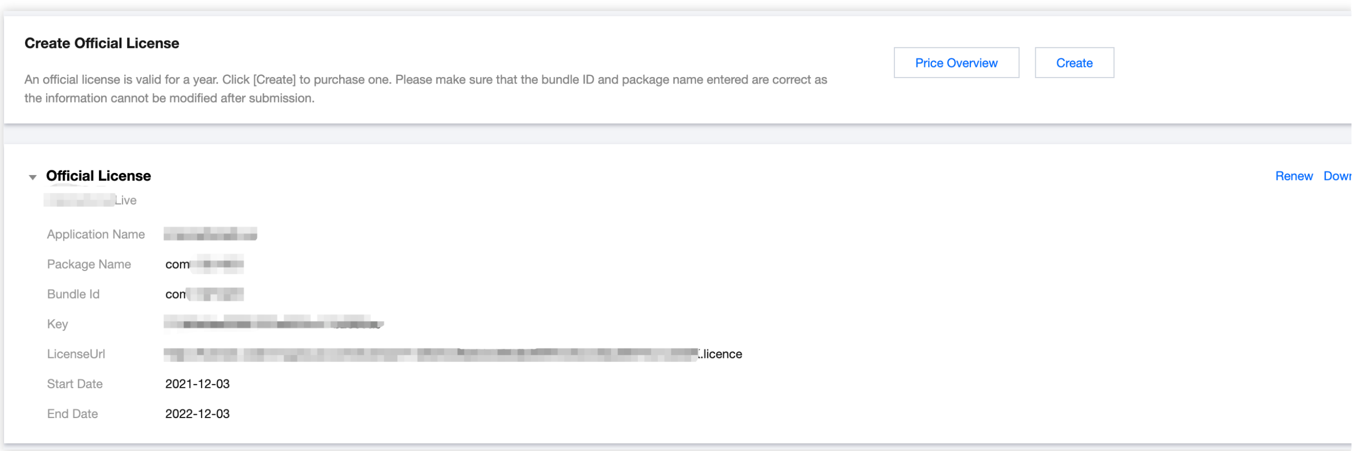
Step 1. Download the SDK

Download the SDK and follow the instructions in SDK Integration to integrate the SDK into your application.

Step 2. Configure License Authorization for SDK

1. Obtain license authorization：

If you have obtained the relevant license authorization， need to Get License URL and License Key in Cloud Live Console.



Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Price Overview
Create

Official License

Renew Dowr

[redacted]Live

Application Name	[redacted]
Package Name	com[redacted]
Bundle Id	com[redacted]
Key	[redacted]
LicenseUrl	[redacted].licence
Start Date	2021-12-03
End Date	2022-12-03

If you have not yet obtained the license authorization, Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:

```
public class MyApplication extends Application {

@Override
public void onCreate() {
super.onCreate();
String licenceURL = ""; // your licence url
String licenceKey = ""; // your licence key
V2TXLivePremier.setEnvironment("GDPR");  // set your environment
V2TXLivePremier.setLicence(this, licenceURL, licenceKey);
V2TXLivePremier.setObserver(new V2TXLivePremierObserver() {
        @Override
        public void onLicenceLoaded(int result, String reason) {
            Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reason);
        }
    });
}
```

Note:

The packageName configured in the license must be the same as the application itself, otherwise the play stream will fail.

Step 3. Create a rendering view

For the player to display video images, you need to add a rendering view in the layout XML file:

```
<com.tencent.rtmp.ui.TXCloudVideoView
            android:id="@+id/video_view"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_centerInParent="true"
```

```
              android:visibility="visible"/>
```

Step 4. Create a player object

The **V2TXLivePlayer** module in the Video Cloud SDK offers live playback capabilities. Use the **setRenderView** API to associate the module with the **video_view** control added to the UI in Step 3.

```
// mPlayerView is the view added in step 1
TXCloudVideoView mView = (TXCloudVideoView) view.findViewById(R.id.video_view);
// Create a player object
V2TXLivePlayer mLivePlayer = new V2TXLivePlayerImpl(mContext);
// Associate the player object with the view
mLivePlayer.setRenderView(mView);
```

Step 5. Start playback

```
String flvUrl = "http://2157.liveplay.myqcloud.com/live/2157_xxxx.flv";
mLivePlayer.startPlay(flvUrl);
```

Step 6. Change the fill mode

view: size and position

You can modify the size and position of video images by adjusting the size and position of the `video_view` control added in Step3.

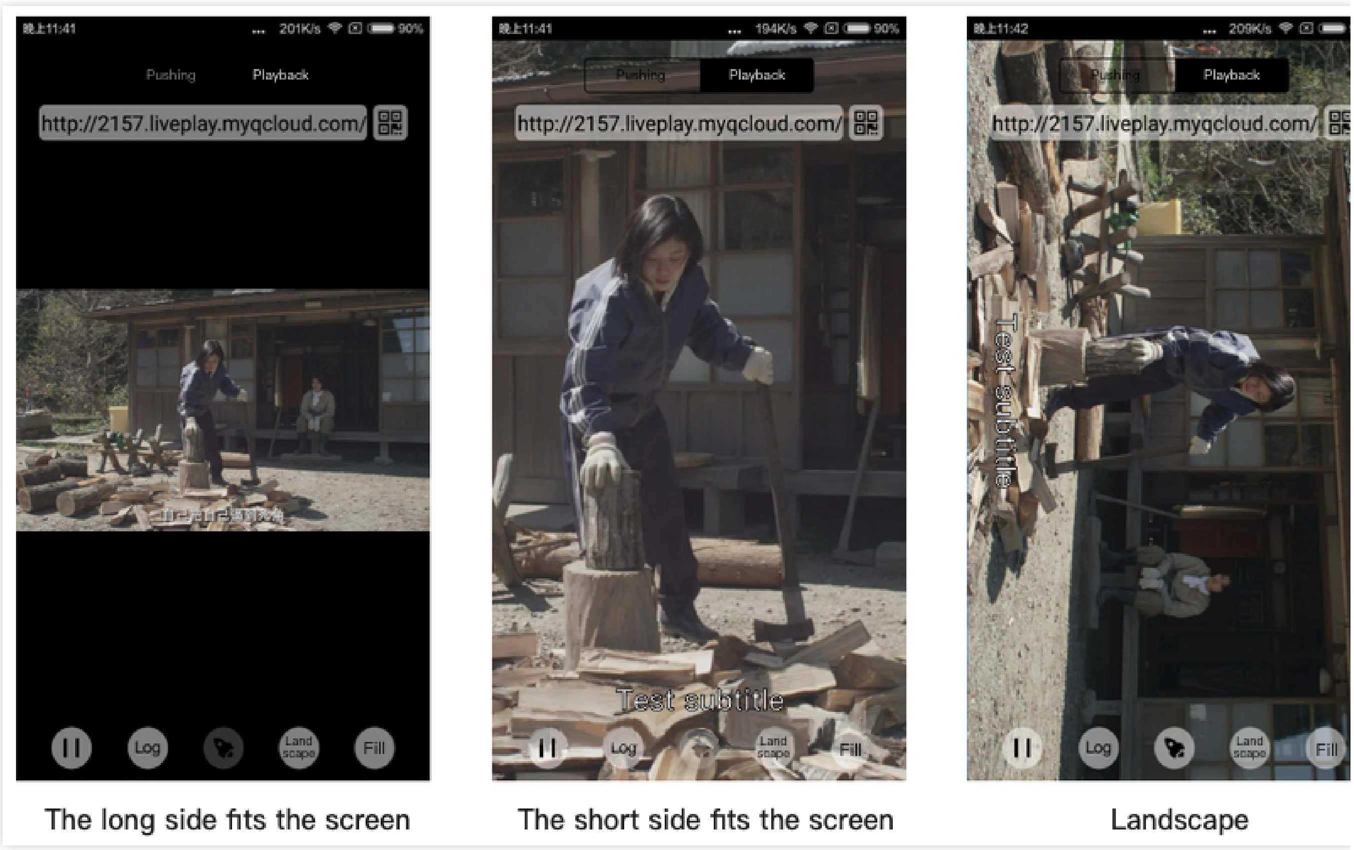
setRenderFillMode: aspect fill or aspect fit

Value	Description
V2TXLiveFillModeFill	Images are scaled to fill the entire screen, and the excess parts are cropped. There are no black bars in this mode, but images may not be displayed in whole.
V2TXLiveFillModeFit	Images are scaled as large as the longer side can go. Neither side exceeds the screen after scaling. Images are centered, and there may be black bars.

setRenderRotation: clockwise rotation of video

Value	Description
V2TXLiveRotation0	Original
V2TXLiveRotation90	Rotate 90 degrees clockwise
V2TXLiveRotation180	Rotate 180 degrees clockwise
V2TXLiveRotation270	Rotate 270 degrees clockwise

```
// Set the fill mode
mLivePlayer.setRenderFillMode(V2TXLiveFillModeFit);
// Set the rotation of video
mLivePlayer.setRenderRotation(V2TXLiveRotation0);
```


Pushing
Playback
http://2157.liveplay.myqcloud.com/
Test subtitle
Log
Land scape
Fill
The long side fits the screen
The short side fits the screen
Landscape

Step 7. Pause playback

Technically speaking, you cannot pause a live playback. In this document, by pausing playback, we mean **freezing video** and **disabling audio**. In the meantime, new video streams continue to be sent to the cloud. When you resume playback, it starts from the time of resumption. This is in contrast to VOD. With VOD, when you pause and resume playback, the player behaves the same way as it does when you pause and resume a local video file.

```
// Pause playback
mLivePlayer.pauseAudio();
mLivePlayer.pauseVideo();
// Resume playback
mLivePlayer.resumeAudio();
mLivePlayer.resumeVideo();
```

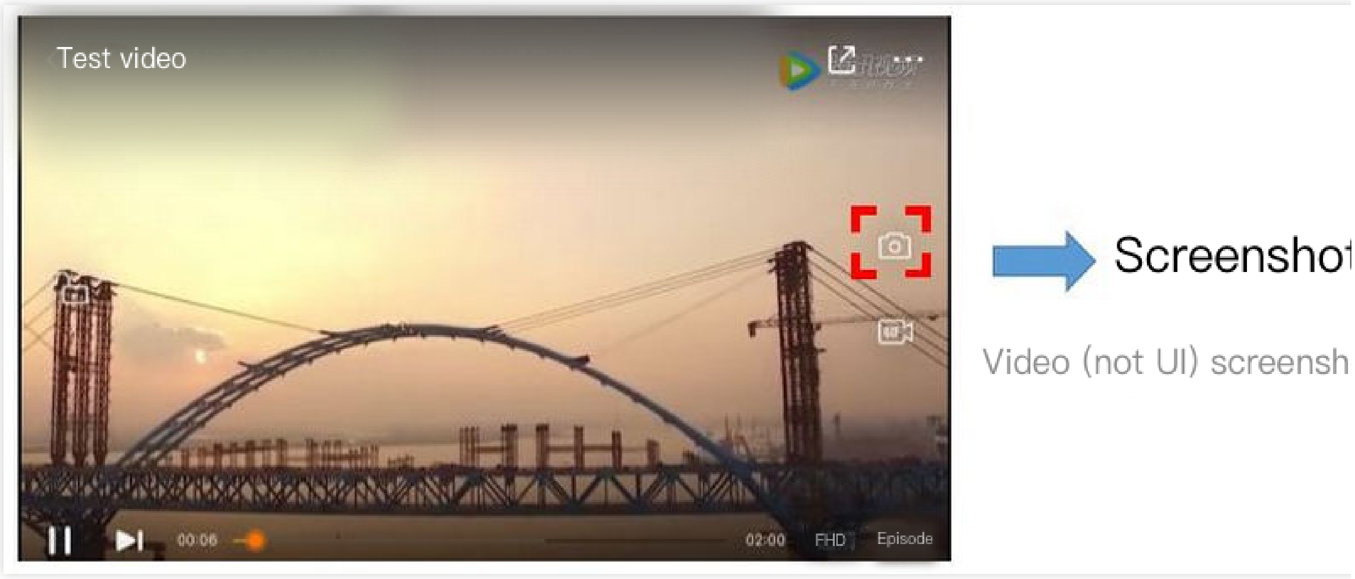
Step 8. Stop playback

Call `stopPlay` to stop playback.

```
mLivePlayer.stopPlay();
```

Step 9. Take a screenshot

Call **snapshot** to take a screenshot of the live video streamed. This method captures a frame of the streamed video. To capture the UI, use the corresponding API of the Android system.



```
mLivePlayer.setObserver(new MyPlayerObserver());
mLivePlayer.snapshot();
// Get the screenshot taken in the onSnapshotComplete callback of MyPlayerObserver
private class MyPlayerObserver extends V2TXLivePlayerObserver  {
    ...
    @Override
    public void onSnapshotComplete(V2TXLivePlayer v2TXLivePlayer, Bitmap bitmap) {
    }
    ...
}
```

Latency Control

The live playback feature of the SDK is not based on FFmpeg, but Tencent Cloud’s proprietary playback engine, which is why the SDK offers better latency control than open-source players do. We provide three latency control modes, which can be used for showrooms, game streaming, and hybrid scenarios.

Comparison of the three modes

Mode	Stutter	Average Latency	Scenario	Remarks
Speedy	More likely than the speedy mode	2 - 3s	Live showroom (Chongding Dahui)	The mode delivers low latency and is suitable for latency-sensitive scenarios.
Smooth	Least likely of the three	≥ 5s	Game streaming (Penguin Esports)	Playback is least likely to stutter in this mode, which makes it suitable for ultra-high-bitrate streaming of games such as PUBG.
Auto	Self-adaptive to network conditions	2 - 8s	Hybrid	The better network conditions at the audience end, the lower the latency.

Code to integrate the three modes

```
// Auto mode
mLivePlayer.setCacheParams(1.0f, 5.0f);
// Speedy mode
mLivePlayer.setCacheParams(1.0f, 1.0f);
// Smooth mode
mLivePlayer.setCacheParams(5.0f, 5.0f);
// Start playback after configuration
```

Note:

For more information on stuttering and latency control, see Video Stutter.

Listening for SDK Events

You can bind a V2TXLivePlayerObserver to your `V2TXLivePlayer` object to receive callback notifications about the player status, playback volume, first audio/video frame, statistics, warning and error messages, etc.

Periodically triggered notification

The onStatisticsUpdate callback notification is triggered every 2 seconds to update you on the player's status in real time. Like a car's dashboard, the callback gives you information about the SDK, such as network conditions and video information.

Parameter	Description

appCpu	CPU usage (%) of the application
systemCpu	CPU usage (%) of the system
width	Video width
height	Video height
fps	Frame rate (fps)
audioBitrate	Audio bitrate (Kbps)
videoBitrate	Video bitrate (Kbps)

The onPlayoutVolumeUpdate callback, which notifies you of the player's volume, works only after you call enableVolumeEvaluation to enable the volume reminder. You can set the interval of the callback by specifying the `intervalMs` parameter of `enableVolumeEvaluation` .

Event-triggered notifications

Other callbacks are triggered when specific events occur.

LEB

Last updated：2024-01-13 15:49:41

LEB Overview

Live Event Broadcasting (LEB) is the ultra-low-latency version of LVB. It features lower latency than traditional streaming protocols and delivers superior playback experience with millisecond latency. It is suitable for scenarios with high requirements on latency, such as online education, sports streaming, and online quizzes.

Note:

The figure above (made using scrcpy) is a comparison of LEB and standard CDN live streaming. The images on the left and in the middle show the playback end of standard CDN live streaming and **LEB**, and the image on the right shows the publishing end.

LVB and LEB are priced differently. For details, please see LVB Billing Overview and LEB Billing Overview.

Strengths

Strength	Description
Playback with millisecond latency	The latency is kept within 1s thanks to the use of UDP, as opposed to 3-5s in traditional live streaming. This, along with excellent instant streaming performance and low stuttering rate, guarantees a superior streaming experience.
Diverse features and smooth migration	LEB integrates a wide range of features including stream publishing, transcoding, recording, screenshot, porn detection, and playback. It allows smooth migration from standard live streaming.
Easy-to-use, secure, and reliable	The use of a standard protocol makes integration easy. You can play live video on Chrome and Safari without installing any plugins. In addition, the playback protocol encrypts video by default for improved security and reliability.

Use cases

Scenario	Description
Sports event	LEB offers ultra-low-latency streaming for sports events. It brings sports content to audience at low latency, allowing audience to learn what’s happening in real time.
E-commerce streaming	Some e-commerce streaming scenarios, for example, online auctions and

	sales promotion, require extremely low latency. LEB’s ability to stream at ultra-low latency ensures that hosts and audience get real-time feedback from each other, improving online shopping experience.
Online classes	LEB can be used for online classes. Its ability to stream at ultra-low latency allows teachers and students to interact with each other as they do in offline classes.
Online quizzes	Due to latency, some online quizzes have to insert extra frames at the audience end to ensure that the host and audience are in sync with each other. This is not necessary if you use LEB, whose ultra-low-latency streaming capability makes sure that the two sides are in sync. It helps you implement online quizzes more easily and deliver smoother experience.
Showrooms	LEB can significantly improve the experience of latency-sensitive interactions such as gift giving in live showrooms.

Tryout

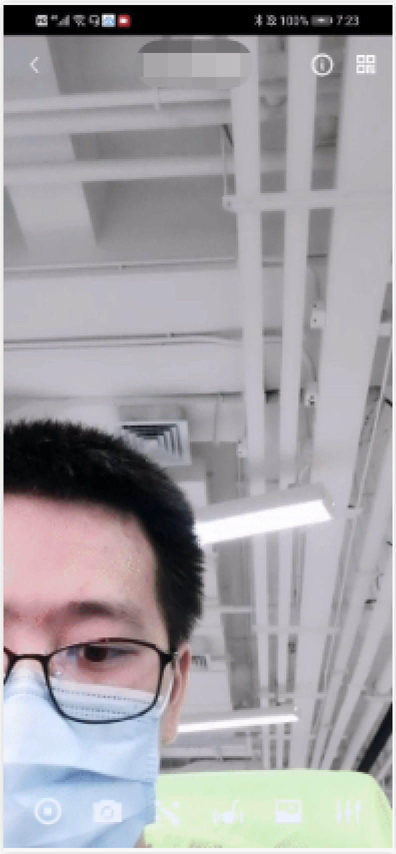
Video Cloud Toolkit is a comprehensive audio-video solution developed by Tencent Cloud that allows you to try out the features of the TRTC, MLVB and UGC SDKs, including the **LEB Player**.

Note:

The demonstration and directions in this document use the demo app for Android as an example. The UI of the app for iOS is slightly different.

Source code and demonstration

Source Code	Demo	Publishing Demonstration (Android)	F
Android			

iOS			

Note:

In addition to the above sample code, regarding frequently asked questions among developers, Tencent Cloud offers a straightforward API example project, which you can use to quickly learn how to use different APIs.

iOS：MLVB-API-Example

Android：MLVB-API-Example

Flutter：Live-API-Example

Publishing

LEB is compatible with LVB, which means you can publish streams using an ordinary publisher and play the streams using LEB.

1. Download and install Video Cloud Toolkit, log in, and go to **MLVB** > **Push (Camera)**.
2. Allow the permissions asked, and tap **Auto-generate** to start publishing streams.
3. If publishing is successful, tap the QR code icon in the top right and select **LEB** to get the playback URL for LEB.
4. During publishing, you can tap the menu icon in the bottom right to apply filters, add background music, switch cameras, etc.

Playback

1. Download and install Video Cloud Toolkit, log in, and go to **Live Broadcast > LEB Player**.
2. Allow the permissions asked, tap the scan button, and scan the LEB playback URL obtained earlier.
3. Playback starts automatically once the QR code is read. During playback, you can tap the menu icon in the bottom right to mute playback or change other settings.

Integration

In the new version of the MLVB SDK, you can use V2TXLivePlayer for LEB and `V2TXLivePusher` for publishing. LEB supports WebRTC protocols and uses the standard extension method. All URLs in LEB start with `webrtc://` .

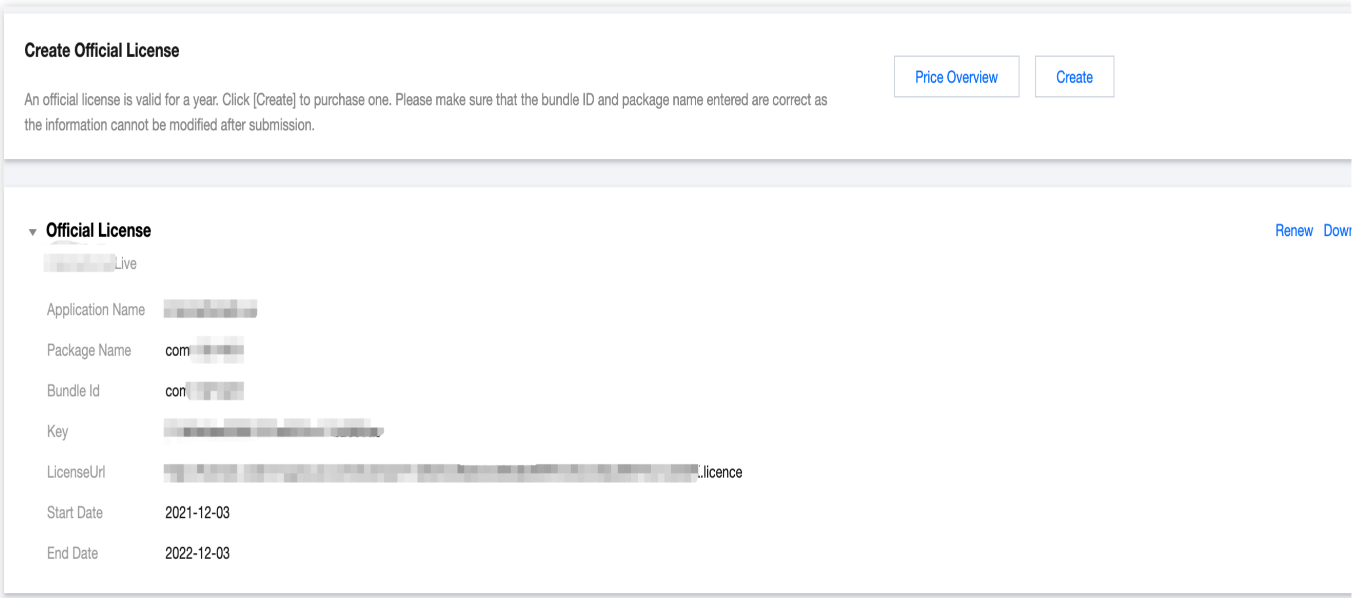
Step 1. Download the SDK

Download LiteAV_All or Professional at SDK Download.

Step 2. Configure License Authorization for SDK

1. Obtain license authorization：

If you have obtained the relevant license authorization， need to Get License URL and License Key in Cloud Live Console


Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

[redacted]Live

Application Name	[redacted]
Package Name	com[redacted]
Bundle Id	con[redacted]
Key	[redacted]
LicenseUrl	[redacted].licence
Start Date	2021-12-03
End Date	2022-12-03

If you have not yet obtained the license authorization， Please reference Adding and Renewing Licenses to make an application.

2. Before your App calls SDK-related functions (it is recommended in the Application class), set the following settings:

```
public class MApplication extends Application {

@Override
```

```
public void onCreate() {
    super.onCreate();
    String licenceURL = ""; // your licence url
    String licenceKey = ""; // your licence key
    V2TXLivePremier.setEnvironment("GDPR");  // set your environment
    V2TXLivePremier.setLicence(this, licenceURL, licenceKey);
    V2TXLivePremier.setObserver(new V2TXLivePremierObserver() {
            @Override
            public void onLicenceLoaded(int result, String reason) {
                Log.i(TAG, "onLicenceLoaded: result:" + result + ", reason:" + reas
            }
        });
}
```

Step 3. Get a playback URL

In live streaming, URLs are needed for both publishing and playback. For instructions on how to get URLs for LEB, please see Live Event Broadcasting (LEB) > Get Playback URL.

All LEB URLs start with `webrtc://` , as in:

```
webrtc://{Domain}/{AppName}/{StreamName}
```

The table below lists the key fields in an LEB URL and their meanings.

Field	Description
webrtc://	Prefix
Domain	Domain name
AppName	Application name, which is `live` by default. It specifies the storage path of a live streaming file.
StreamName	Stream name, which is the unique identifier of a stream

Note:

To publish streams, please see Publishing from Camera or Publishing from Screen.

Step 4. Start LEB

You can use a `V2TXLivePlayer` object for LEB. For details, see the code below (make sure that you pass in the correct URL).

Sample code

Android

iOS

```
// Create a V2TXLivePlayer object
V2TXLivePlayer player = new V2TXLivePlayerImpl(mContext);
player.setObserver(new MyPlayerObserver(playerView));
player.setRenderView(mSurfaceView);
// Pass in the low-latency playback URL to start playback
player.startPlay("webrtc://{Domain}/{AppName}/{StreamName}");
```

```
V2TXLivePlayer *player = [[V2TXLivePlayer alloc] init];
[player setObserver:self];
[player setRenderView:videoView];
[player startPlay:@"webrtc://{Domain}/{AppName}/{StreamName}"];
```

Flutter
LVB

Last updated：2024-01-13 15:49:41

Basics

This document describes the live playback feature of Live Flutter Plugin.

Live streaming overview

In **live streaming**, the video source is pushed by the host in real time. When the host stops pushing the source, the player will also stop playing the video. Because the live stream is played back in real time, no progress bar will be displayed in the player during the playback.

Supported protocols

Common live streaming protocols are as listed below. We recommend you use an FLV-based live streaming URL that starts with `http` and ends with `.flv` for LVB. For LEB, we recommend you use the WebRTC protocol. For more information, see LEB.

Protocol	Advantage	Disadvantage	Playback Latency
HLS	Mature, well adapted to high-concurrency scenarios	SDK integration is required	3–5 seconds
FLV	Mature, well adapted to high-concurrency scenarios	SDK integration is required.	2-3 seconds
RTMP	Relatively low latency	Poor performance in high-concurrency scenarios	1-3 seconds
WebRTC	Lowest latency	SDK integration is required	< 1 second

Note:
LVB and LEB are priced differently. For details, see LVB Billing Overview and LEB Billing Overview.

Notes

The SDK **does not impose any restrictions on the sources of playback URLs**, which means you can use it to play both Tencent Cloud and non-Tencent Cloud URLs. However, the player of the SDK supports only live streaming URLs in FLV, RTMP, HLS (M3U8), and WebRTC formats and VOD URLs in MP4, HLS (M3U8), and FLV formats.

Sample Code

Tencent Cloud offers an easy-to-understand API example project to help you quickly learn how to use different APIs.

Platform	GitHub Address
iOS	GitHub
Android	GitHub
Flutter	live_flutter_plugin

Getting Started

1. Set dependencies

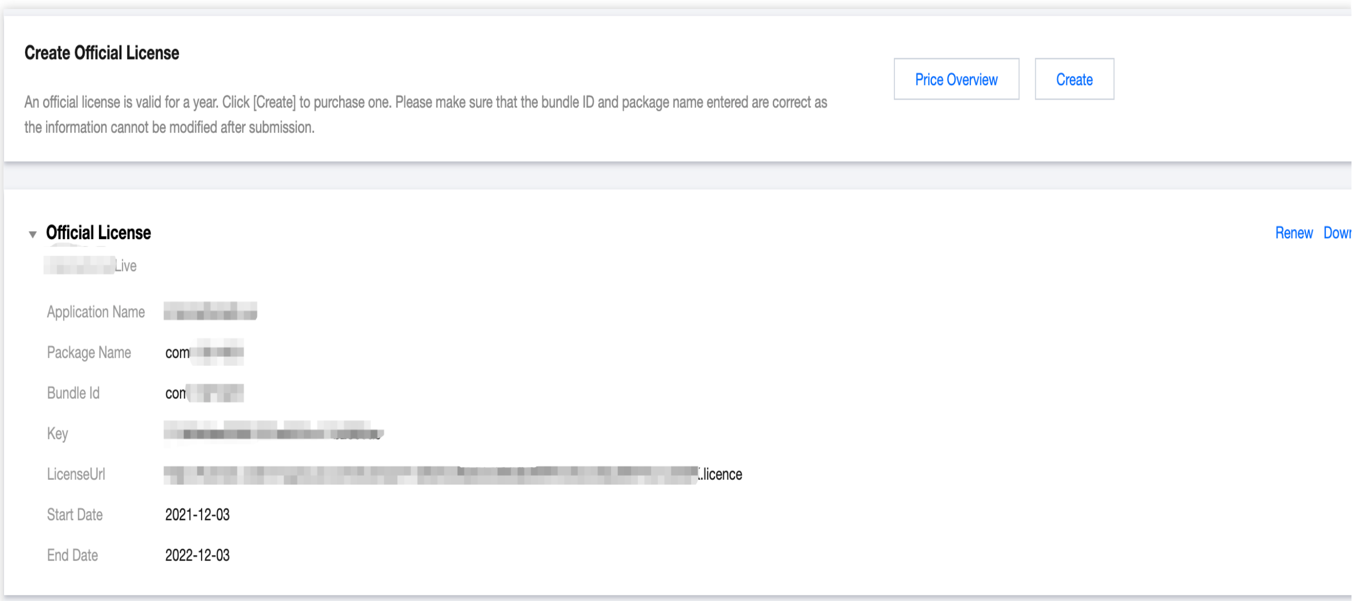
Integrate `live_flutter_plugin` into your application as instructed in SDK Integration Guide.

```
dependencies:
  live_flutter_plugin: latest version number
```

2. Configure a license for the SDK

1. Get the license:

If you have the required license, get the license URL and key in the CSS console.


Create Official License

An official license is valid for a year. Click [Create] to purchase one. Please make sure that the bundle ID and package name entered are correct as the information cannot be modified after submission.

Official License

[redacted]Live

Application Name	[redacted]
Package Name	com[redacted]
Bundle Id	con[redacted]
Key	[redacted]
LicenseUrl	[redacted].licence
Start Date	2021-12-03
End Date	2022-12-03

If you don't have the required license, apply for a license as instructed in New License and Renewal.

2. Before your application calls features of `live_flutter_plugin` , complete the following configuration:

```
import 'package:live_flutter_plugin/v2_tx_live_premier.dart';

 /// Tencent Cloud license management page (https://console.tencentcloud.com/live/l
setupLicense() {
  // The license URL of the current application
  var LICENSEURL = "";
  // The license key of the current application
  var LICENSEURLKEY = "";
  V2TXLivePremier.setLicence(LICENSEURL, LICENSEURLKEY);
}
```

Note:

The `packageName/BundleId` configured in the license must be the same as that of the application; otherwise, playback will fail.

3. Create the player

The `V2TXLivePlayer` module in the SDK offers live playback capabilities.

```
import 'package:live_flutter_plugin/v2_tx_live_player.dart';
/// Initialize `V2TXLivePlayer`
initPlayer() {
  _livePlayer = V2TXLivePlayer();
  _livePlayer.addListener(onPlayerObserver);
}
```

4. Render the view

In Flutter, you need to depend on `v2_tx_live_video_widget` to create a video rendering view for the player to display video images on.

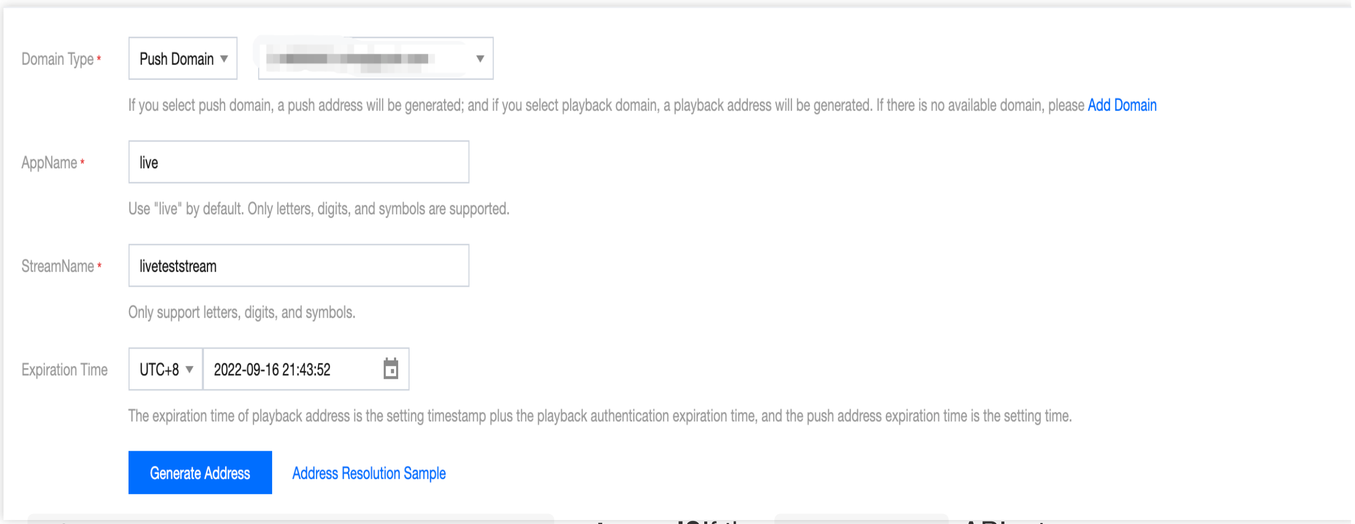
```
import 'package:live_flutter_plugin/widget/v2_tx_live_video_widget.dart';

/// The video rendering view widget
Widget renderView() {
  return V2TXLiveVideoWidget(
    onViewCreated: (viewId) async {
      // Set the video rendering view
      _livePlayer.setRenderViewID(_renderViewId);
    },
  );
}
```

5. Start playback

```
/// Start pulling the stream
startPlay() async {
  // Generate the `url` (RTMP/TRTC/LEB)
  var url = ""
  // Start pulling the stream
  await _livePlayer?.startPlay(url);
}
```

How do I get a valid stream push URL?

Activate CSS. In the **CSS console**, go to **Auxiliary Tools** > **Address Generator** to generate a stream push URL. For more information, see Publishing/Playback URL.



Domain Type *	Push Domain
	If you select push domain, a push address will be generated; and if you select playback domain, a playback address will be generated. If there is no available domain, please Add Domain
AppName *	live
	Use "live" by default. Only letters, digits, and symbols are supported.
StreamName *	liveteststream
	Only support letters, digits, and symbols.
Expiration Time	UTC+8 2022-09-16 21:43:52
	The expiration time of playback address is the setting timestamp plus the playback authentication expiration time, and the push address expiration time is the setting time.

Why is `V2TXLIVE_ERROR_INVALID_LICENSE` returned?If the `startPush` API returns `V2TXLIVE_ERROR_INVALID_LICENSE` , it means your license verification failed. Please check your configuration against Step 2. Configure a license for the SDK.

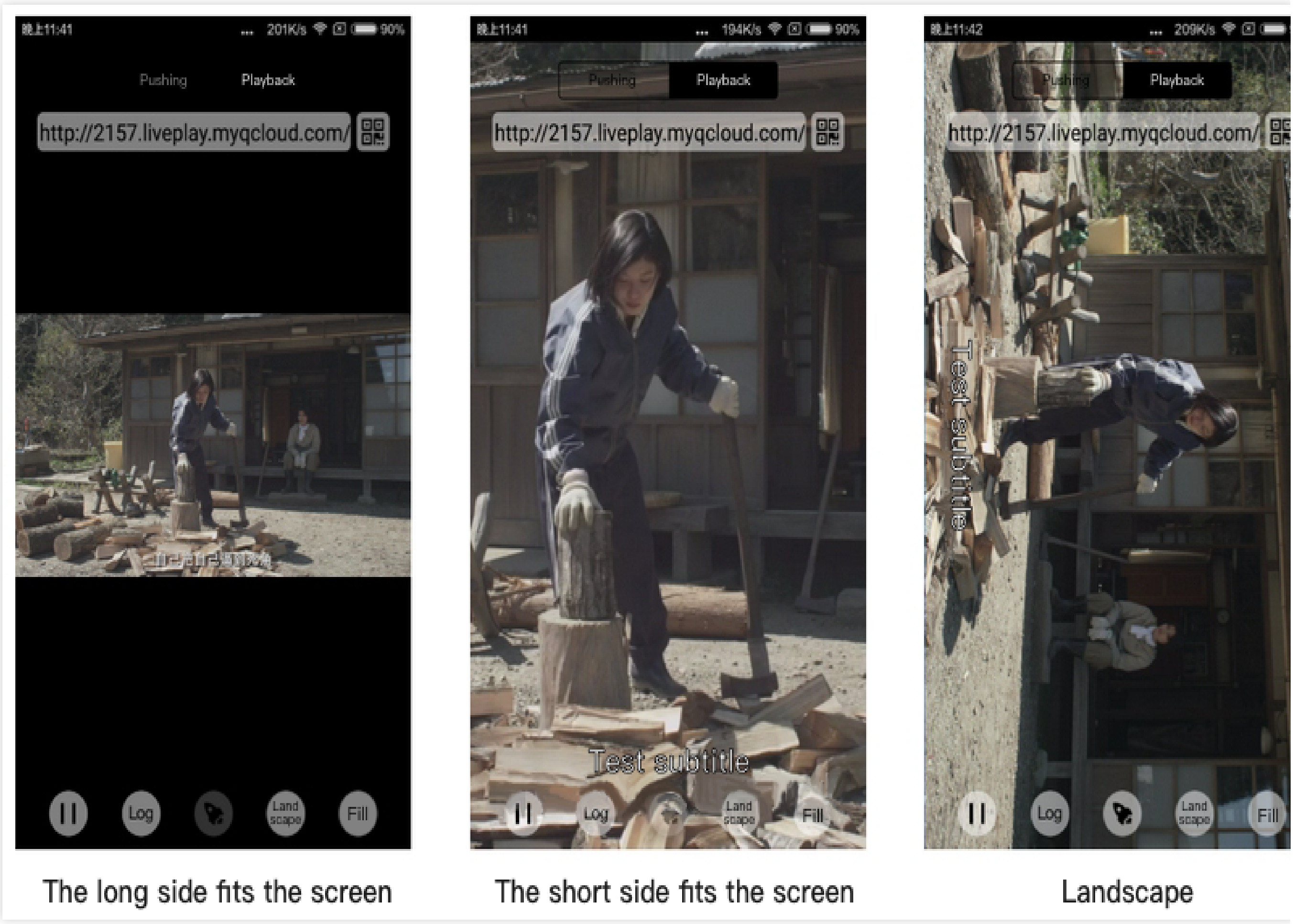
6. Adjust the image

setRenderFillMode: Fill or fit

Value	Description
V2TXLiveFillModeFill	Images are scaled to fill the entire screen, and the excess parts are cropped. There are no black bars in this mode, but images may not be displayed entirely.
V2TXLiveFillModeFit	Images are scaled so that the long side of the video fits the screen. Neither side exceeds the screen after scaling. Images are centered, and there may be black bars visible.

setRenderRotation: Clockwise rotation of video

Value	Description
V2TXLiveRotation0	No rotation
V2TXLiveRotation90	Rotate 90 degrees clockwise
V2TXLiveRotation180	Rotate 180 degrees
V2TXLiveRotation270	Rotate 270 degrees clockwise


Pushing
Playback
http://2157.liveplay.myqcloud.com/
Test subtitle
Log
Land scape
Fill
The long side fits the screen
The short side fits the screen
Landscape

7. Pause playback

Technically speaking, you cannot pause a live playback. In this document, pausing live playback refers to **freezing the video** and **disabling the audio**. While the video is frozen, new video streams continue to be sent to the cloud. When you resume playback, it resumes playing the current latest stream. This is different from pausing and resuming playback in VOD, in which the player behaves the same way as it does when you pause and resume a local video file.

```
// Pause playback
_livePlayer.pauseAudio();
_livePlayer.pauseVideo();
// Resume playback
_livePlayer.resumeAudio();
_livePlayer.resumeVideo();
```

8. Stop playback

```
// Stop playback
_livePlayer.stopPlay();
```

Latency Control

The live playback feature of the SDK is not based on FFmpeg, but Tencent Cloud's proprietary playback engine, which is why the SDK offers better latency control than open-source players do. We provide three latency control modes, which can be used for live showrooms, game streaming, and hybrid scenarios.

Comparison of control modes

Mode	Stutter	Average Latency	Scenario	Remarks
Speedy	Relatively high	2-3s	Live showroom	Has better latency control and is suitable for scenarios that require a low latency.
Smooth	Low	>= 5s	Game streaming	Suitable for game live streaming scenarios with a high bitrate.
Auto	Adapts to network conditions	2-8s	Hybrid	The better the network conditions at the audience end, the lower the latency.

Code to integrate the three modes

```
// Auto mode
_txLivePlayer.setCacheParams(1, 5);
// Speedy mode
_txLivePlayer.setCacheParams(1, 1);
// Smooth mode
_txLivePlayer.setCacheParams(5, 5);

// Start playback after configuration
```

Note:

For more information on stutter and latency control, see Video Stutter.

Listening for SDK Events

You can bind a V2TXLivePlayerObserver to `V2TXLivePlayer` . Then, all internal SDK status information, such as player status, playback volume level, reception of the first audio/video frame, statistical data, warnings, and errors, will be notified to you through corresponding callbacks.

Periodically triggered notification

The onStatisticsUpdate notification is triggered once every 2 seconds to provide real-time feedback on the current player status. It can act as a dashboard to inform you of what is happening inside the SDK so you can better understand the current network conditions and video information.

Parameter	Description
appCpu	CPU usage (%) of the application
systemCpu	CPU usage (%) of the system
width	Video width
height	Video height
fps	Frame rate (fps)
audioBitrate	Audio bitrate (Kbps)
videoBitrate	Video bitrate (Kbps)

onPlayoutVolumeUpdate is the callback for the player volume level. It will be returned only when you call enableVolumeEvaluation to enable the prompt for the player volume level. The callback interval is the same as that specified by the `intervalMs` parameter in `enableVolumeEvaluation` .

Event-triggered notifications

Other callbacks are triggered when specific events occur.