直播 SDK

准备工作

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

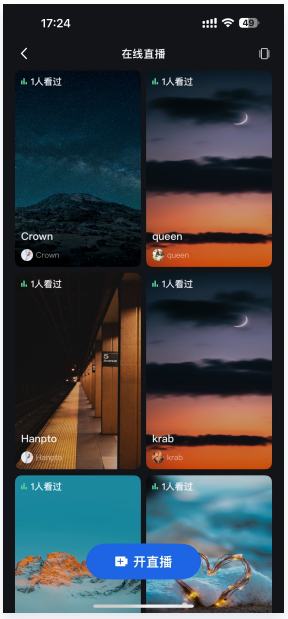
准备工作
准备工作（Android）

最近更新时间：2026-03-03 18:01:02

功能预览

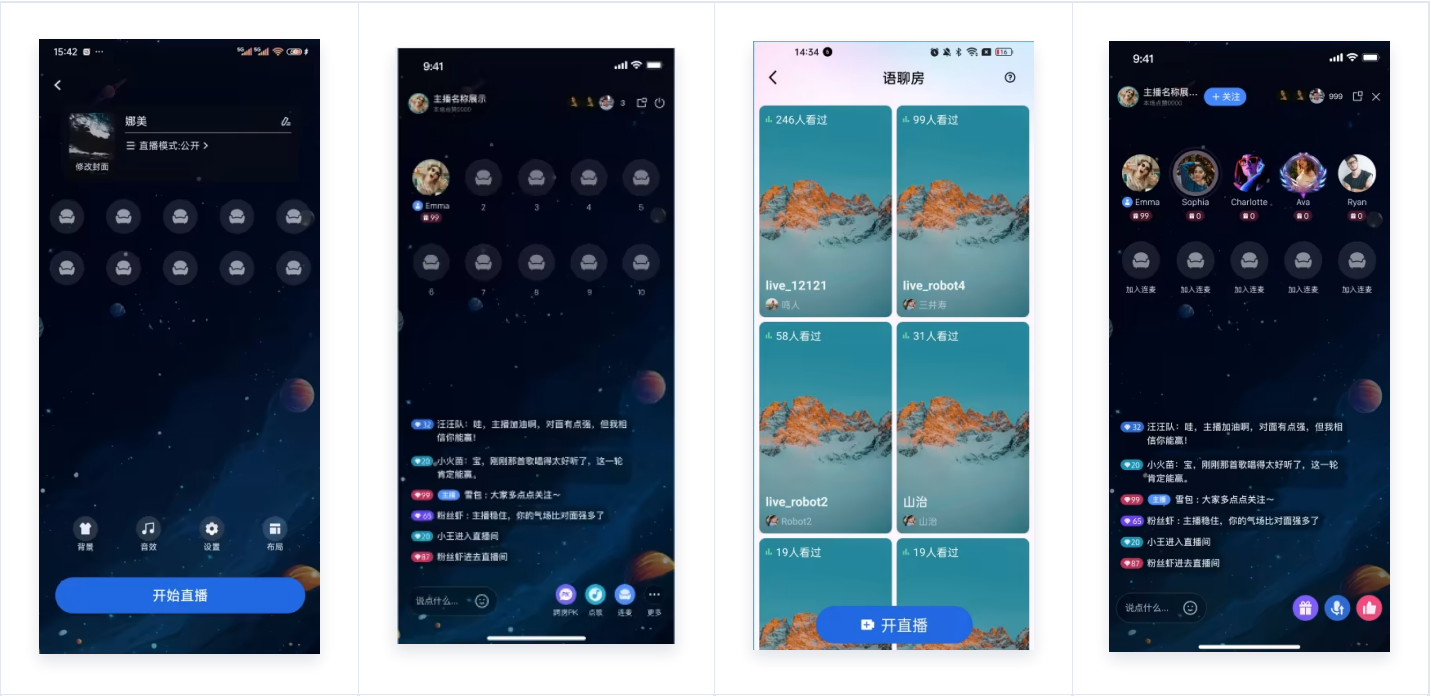
`TUILiveKit` 是一个功能全面的直播组件，集成后可快速实现以下场景功能。

视频直播

主播准备页	主播开播页	直播列表	观众观看页
			

语聊房

主播准备页	主播开播页	直播列表	观众观看页



准备工作

开通服务

在使用 TUILiveKit 前，请先参考 开通服务，领取 TUILiveKit 体验版或者开通付费版。

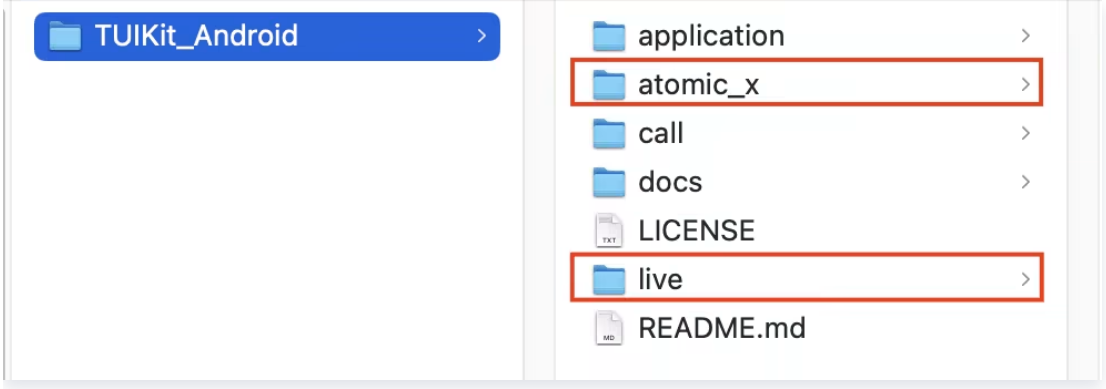
环境要求

- Android Studio Arctic Fox (2020.3.1) 及以上版本。
- Gradle 7.0 及以上的版本。
- Android 5.0 及以上的手机设备。

代码集成

步骤 1：下载 TUILiveKit 组件

在 GitHub 中克隆/下载代码，然后将 `live` 子目录和 `atomic_x` 子目录拷贝到您当前 `Android` 项目的 `app` 文件夹同级目录中。



步骤 2：工程配置

1. 配置 JitPack 仓库

在工程根目录下的 `settings.gradle.kts` 或 `settings.gradle` 文件中，添加 JitPack 仓库地址，项目中播放礼物 SVG 动画的第三方库托管在 JitPack 仓库中，需要从 JitPack 仓库中查找依赖项。

settings.gradle.kts

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()

        // 添加 JitPack 仓库地址
        maven { url = uri("https://jitpack.io") }
    }
}
```

settings.gradle

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()

        // 添加 JitPack 仓库地址
        maven { url 'https://jitpack.io' }
    }
}
```

2. 导入 TUILiveKit 组件

在工程根目录下的 `settings.gradle.kts` 或 `settings.gradle` 文件中，增加以下代码以导入 tuilivekit 组件。

settings.gradle.kts

```
include(":tuilivekit")
project(":tuilivekit").projectDir = File(settingsDir, "live/tuilivekit")

include(":atomic")
project(":atomic").projectDir = File(settingsDir, "atomic_x")
```

settings.gradle

```
include ':tuilivekit'
project(':tuilivekit').projectDir = new File(settingsDir,
"live/tuilivekit")

include(":atomic")
project(':atomic').projectDir = new File(settingsDir, "atomic_x")
```

3. 添加组件依赖

在 `app` 目录下找到 `build.gradle.kts`（或 `build.gradle`）文件，并在文件中增加以下代码，该步骤的作用是声明当前 App 对新加入的 **tuilivekit** 组件的依赖。

build.gradle.kts

```
dependencies {
    // 添加 tuilivekit 依赖
    api(project(":tuilivekit"))
}
```

build.gradle

```
dependencies {
    // 添加 tuilivekit 依赖
    api project(':tuilivekit')
}
```

> **说明：**
> TUILiveKit 内部已默认依赖 TRTC SDK、IM SDK 等公共库，您无需单独配置。

4. 配置混淆规则

由于 SDK 内部使用了 Java 反射，请在 `proguard-rules.pro` 文件中添加以下代码，将相关类加入不混淆名单，以确保功能正常运行。

```
-keep class com.tencent.** { *; }
-keep class com.tencent.beacon.** { *; }
-keep class com.tencent.cloud.iai.lib.** { *; }
-keep class com.tencent.qimei.** { *; }
-keep class com.tencent.xmagic.** { *; }
-keep class com.trtc.uikit.livekit.component.gift.store.model.** { *;
}
-keep class com.trtc.uikit.livekit.livestreamcore.** { *; }
-keep class com.tcmediax.** { *; }

# Google 序列化/反序列化框架 Gson 库相关混淆
-keep class com.google.gson.** { *; }

# 美颜特效相关混淆
-keep class androidx.exifinterface.** { *; }
-keep class com.gyailib.** { *; }
-keep class org.extra.** { *; }
-keep class org.libpag.** { *; }
-keep class org.light.** { *; }

# 播放礼物 SVG 动画相关混淆
-keep class com.opensource.svgaplayer.proto.** { *; }
-keep class com.squareup.wire.** { *; }
```

5. 修改 AndroidManifest.xml

为了防止编译时 `AndroidManifest` 合并过程中产生属性冲突，您需要在 `app/src/main/AndroidManifest.xml` 文件的 `<application>` 节点中，添加 `tools:replace="android:allowBackup"` 和 `android:allowBackup="false"`，用以覆盖组件内的设置。

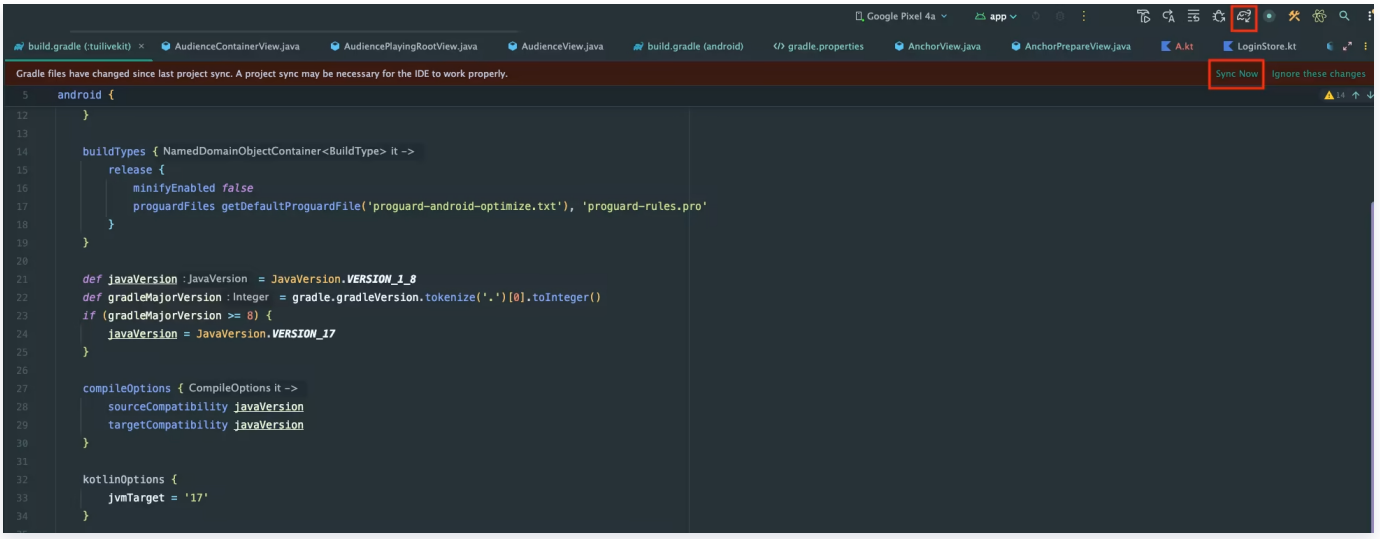
```
<application
    ...

    // 添加如下配置覆盖 依赖的 sdk 中的配置
    android:allowBackup="false"
```

```
    tools:replace="android:allowBackup">
```

6. 完成项目同步

在您完成上述步骤后，通常情况下 Android Studio 会自动为您弹出 **Sync Now** 按钮，您需要点击该同步按钮完成项目的同步，若 IDE 没有自动弹出同步按钮，请您手动点击工具栏中的同步按钮进行项目的同步，同步后 IDE 会完成项目的构建配置和索引等工作，您就可以在您的项目中正常使用 TUILiveKit 组件了。



完成登录

代码集成完成后，您需要完成登录。这是**使用 TUILiveKit 的关键步骤**，因为只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

> **说明：**
> 在示例代码中，直接进行了登录接口的调用。但在实际项目场景下，强烈推荐**您在完成自己的用户身份验证等相关登录操作后，再调用 TUILiveKit 的登录服务**。这样可以避免因过早调用登录服务，导致业务逻辑混乱或数据不一致的问题，同时也能更好地适配您项目中现有的用户管理和权限控制体系。

Kotlin

```
// 1.导入依赖
import com.tencent.qcloud.tuicore.TUILogin

// 2.调用登录接口，推荐您在自己登录业务完成后再调用 TUILogin 登录
TUILogin.login(applicationContext,
    1400000001,       // 请替换为开通服务控制台的 SDKAppID
    "denny",          // 请替换为您的 UserID
    "xxxxxxxxxxx",    // 您可以在控制台中计算一个 UserSig 并填在这个位置
```

```
    object : TUICallback() {
        override fun onSuccess() {
            Log.i(TAG, "login success")
        }

        override fun onError(errorCode: Int, errorMessage: String) {
            Log.e(TAG, "login failed, errorCode: $errorCode
msg:$errorMessage")
        }
    })
```

Java

```
// 1.导入依赖
import com.tencent.qcloud.tuicore.TUILogin

// 2.调用登录接口，推荐您在自己登录业务完成后再调用 TUILogin 登录
TUILogin.login(context,
    1400000001,      // 请替换为开通服务控制台的 SDKAppID
    "denny",         // 请替换为您的 UserID
    "xxxxxxxxxxx",   // 您可以在控制台中计算一个 UserSig 并填在这个位置
    new TUICallback() {
    @Override
    public void onSuccess() {
        Log.i(TAG, "login success");
    }

    @Override
    public void onError(int errorCode, String errorMessage) {
        Log.e(TAG, "login failed, errorCode: " + errorCode + " msg:" +
errorMessage);
    }
});
```

登录接口参数说明

参数	类型	说明

SDKAppID	Int	从 TRTC 控制台 > 应用管理 获取。
UserID	String	当前用户的唯一 ID，仅包含英文字母、数字、连字符和下划线。
userSig	String	用于腾讯云鉴权的票据。请注意： • 开发环境：您可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 UserSig 或者通过 UserSig 辅助工具 生成临时的 UserSig。 • 生产环境：为了防止密钥泄露，请务必采用服务端生成 UserSig 的方式。详细信息请参见 服务端生成 UserSig 。 更多信息请参见 如何计算及使用 UserSig 。

登录异常状态处理【可选】

`TUILogin` 提供了登录状态回调机制，方便您处理可能出现的登录异常情况，主要包括 **“被踢下线”** 和 **“签名过期”** 这两种异常状态的回调：

- **被踢下线**：用户在线情况下被踢，`IM SDK` 会通过 `onKickedOffline` 回调通知给您，此时可以在 UI 提示用户，并调用 `TUILogin.login` 重新登录。
- **签名过期**：用户在线期间收到 `onUserSigExpired` 回调，说明您之前给该用户签发的 userSig 已经过期了，这个时候如果当前用户在您后台的登录态依然有效，您可以让您的 app 向您的后台请求新的 userSig，并调用 `TUILogin.login` 续签登录态。

kotlin

```
class YourLoginService : TUILoginListener() {

    // 监听登录状态回调
    fun addObserver() {
        TUILogin.addLoginListener(this)
    }

    // 取消监听登录状态回调
    fun removeObserver() {
        TUILogin.removeLoginListener(this)
    }

    // 用户被踢下线回调
    override fun onKickedOffline() {
        // 您的业务代码：UI 交互提示用户，然后重新登录
    }
```

```
    // 用户签名过期回调
    override fun onUserSigExpired() {
        // 您的业务代码：如果当前用户在您后台的登录态依然有效，您可以让您的 app 向您的后台请求新的 userSig，并调用 TUILogin.login 续签登录态。
    }
}
```

Java

```
class YourLoginService extends TUILoginListener {

    // 监听登录状态回调
    public void addObserver() {
        TUILogin.addLoginListener(this);
    }

    // 取消监听登录状态回调
    public void removeObserver() {
        TUILogin.removeLoginListener(this);
    }

    // 用户被踢下线回调
    @Override
    public void onKickedOffline() {
        // 您的业务代码：UI 交互提示用户，然后重新登录
    }

    // 用户签名过期回调
    @Override
    public void onUserSigExpired() {
        // 您的业务代码：如果当前用户在您后台的登录态依然有效，您可以让您的 app 向您的后台请求新的 userSig，并调用 TUILogin.login 续签登录态。
    }
}
```

下一步

恭喜您，现在您已经成功集成了**直播**组件并完成了登录。接下来，您可以根据您的业务场景实现**主播开播**、**观众观看**、**直播列表**等功能。

视频直播场景

功能	描述	集成指引
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
观众观看	实现观众进入主播的直播间后观看直播，实现观众连麦、直播间信息、在线观众、弹幕显示等功能。	观众观看
直播列表	展示直播列表界面和功能，包含直播列表、房间信息展示功能。	直播列表

语聊房场景

功能	描述	集成指引
主播开播	主播创建语聊房全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
观众观看	观众进入语聊房后收听，实现上麦、弹幕显示等功能。	观众观看
直播列表	展示语聊房列表界面和功能，包含语聊房列表、房间信息展示功能。	直播列表

常见问题

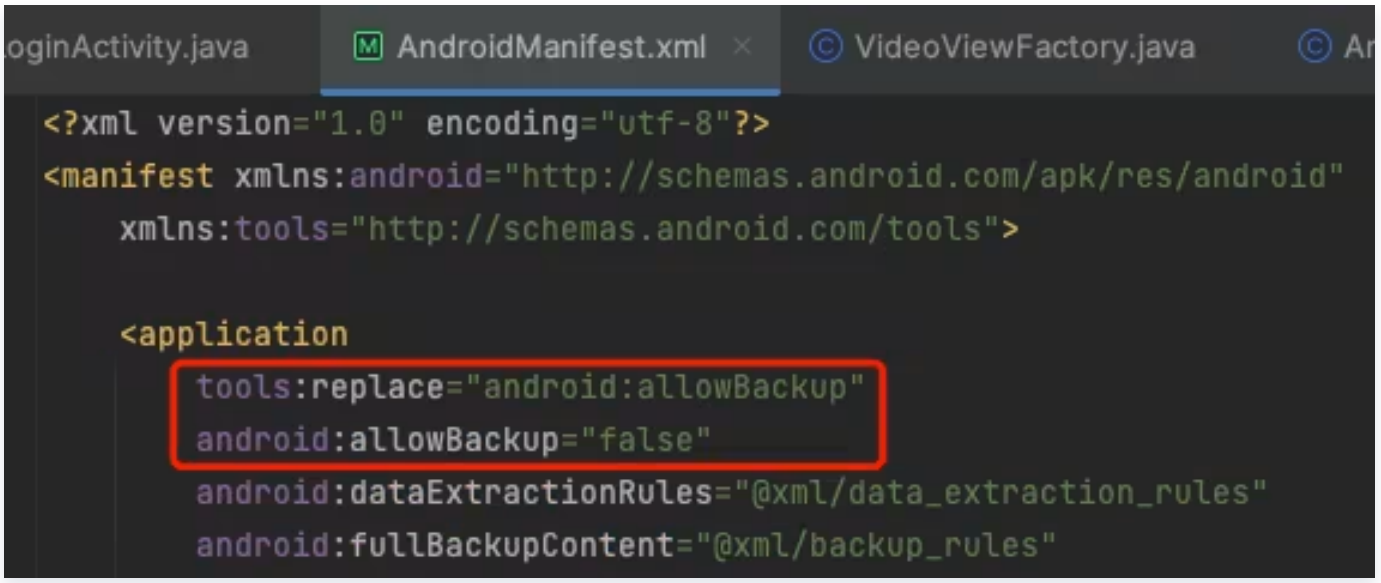
每次进房都需要调用登录吗?

不需要。通常您只需要完成一次 `TUILogin.login` 调用即可，我们建议您将 `TUILogin.login` 和 `TUILogin.logout` 与自己的登录业务关联。

集成代码后产生如下图所示编译报错 allowBackup 异常，如何处理?

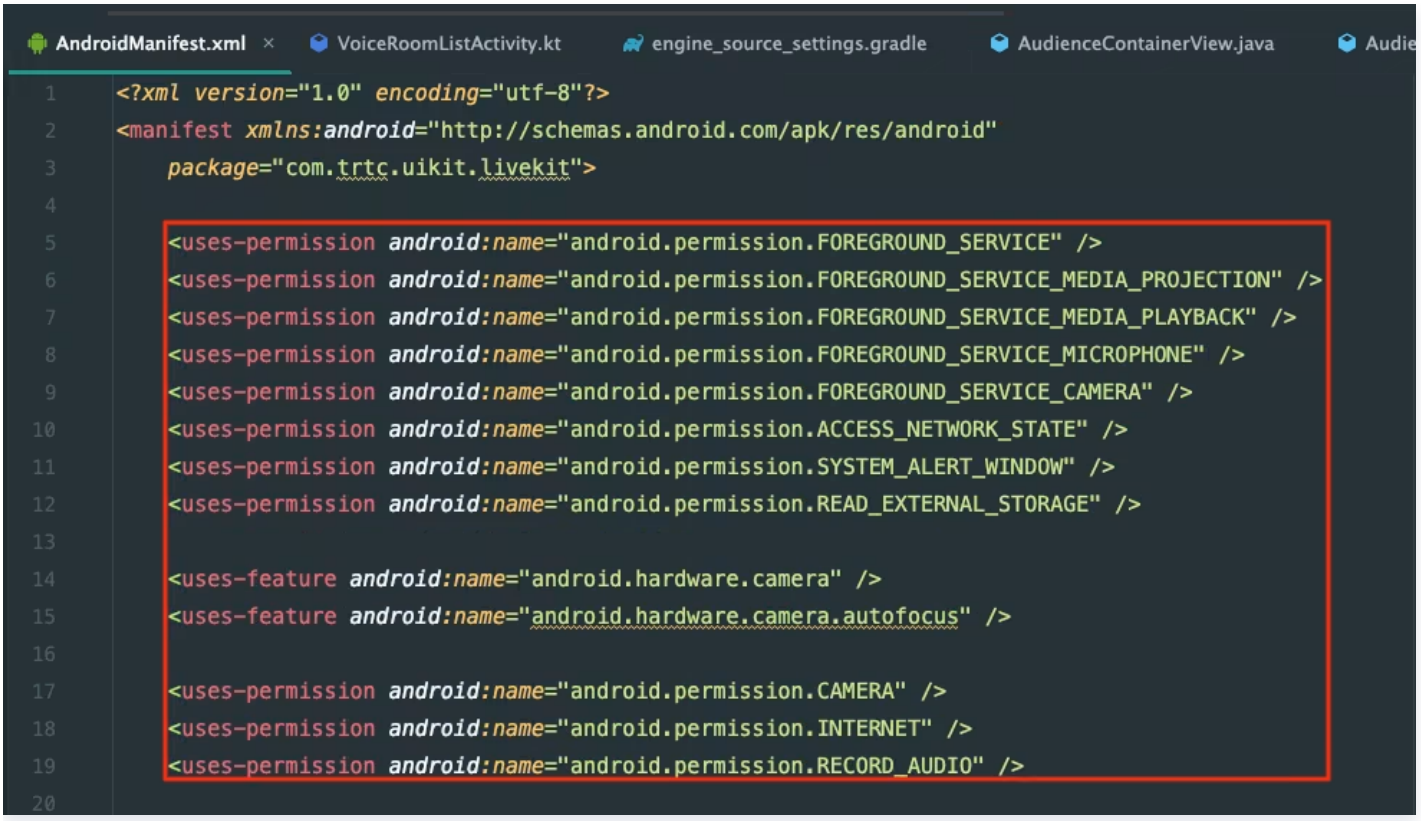
```
Manifest merger failed : Attribute application@allowBackup value=(false) from AndroidManifest.xml:7:9-36
    is also present at [com.github.yyued:SVGAPlayer-Android:2.6.1] AndroidManifest.xml:12:9-35 value=(true).
    Suggestion: add 'tools:replace="android:allowBackup"' to <application> element at AndroidManifest.xml:5:5-53:19 to override.
```

- **问题原因：**多个模块的 `AndroidManifest.xml` 中都配置了 `allowBackup` 属性，造成冲突。
- **解决方法：**您可以在您工程的 `AndroidManifest.xml` 文件中删除 `allowBackup` 属性或将该属性改为 `false`，表示关闭备份和恢复功能；并在 `AndroidManifest.xml` 文件的 application 节点中添加 `tools:replace="android:allowBackup"` 表示覆盖其他模块的设置，使用您自己的设置。修复示例如图所示：



集成代码后还需要添加摄像头麦克风等权限的声明吗?

不需要，TUILiveKit 中已经内置了摄像头麦克风等权限的声明，在接入过程中您无需再关心这些权限的声明。



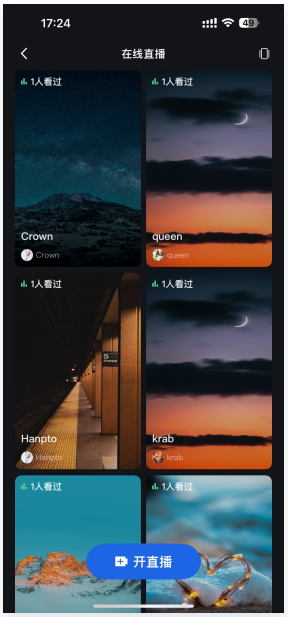
准备工作（iOS）

最近更新时间：2026-03-03 18:01:03

功能预览

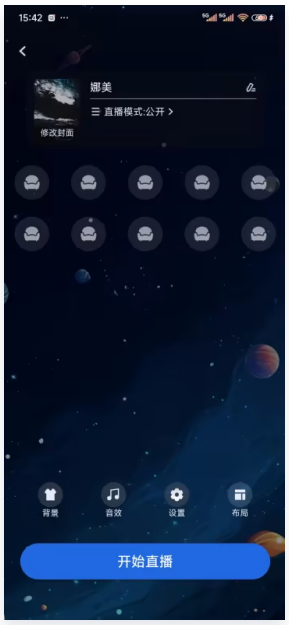
`TUILiveKit` 是一个功能全面的直播组件，集成后可快速实现以下场景功能。

视频直播

主播准备页	主播开播页	直播列表	观众观看页
			

语聊房

主播准备页	主播开播页	直播列表	观众观看页



准备工作

开通服务

在使用 TUILiveKit 前，请先参考 开通服务，领取 TUILiveKit 体验版或者开通付费版。

环境要求

- **Xcode**：需使用 Xcode 15 或更高版本。
- **iOS 系统**：支持 iOS 13.0 或更高版本的设备。
- **CocoaPods 环境**：已安装 CocoaPods 环境。如果您尚未安装，请参见 CocoaPods官网安装，或按以下步骤操作：
  - **使用 gem 安装 CocoaPods**：在终端中执行 `sudo gem install cocoapods` 命令进行安装。

> **提示：**
> `sudo gem install cocoapods` 安装过程中可能需要输入电脑密码，按提示输入管理员密码即可。

代码集成

步骤 1：通过 CocoaPods 导入组件

1. **添加 Pod 依赖：**
   - **若项目已有 Podfile 文件。**

     在您项目的 `Podfile` 文件中添加 `pod 'TUILiveKit'` 依赖。例如：

```
  target 'YourProjectTarget' do
    # 其他已有的 Pod 依赖...
    # 添加 pod 'TUILiveKit' 依赖
    pod 'TUILiveKit'

  end
```

- **若项目没有 Podfile 文件。**

  在终端中通过 `cd` 命令切换到您的 `.xcodeproj` 目录下，然后执行 `pod init` 命令创建 `Podfile` 文件，创建完成后，在您的 `Podfile` 文件中添加 `pod 'TUILiveKit'` 依赖。例如：

```
// 如果您的项目目录是 /Users/yourusername/Projects/YourProject

// 1. cd 到您的 .xcodeproj 工程目录下
cd /Users/yourusername/Projects/YourProject

// 2. 执行 pod init，此命令运行完后，会在您的工程目录下生成一个 Podfile 文件。
pod init

// 3. 在生成的 Podfile 文件中添加 pod 'TUILiveKit' 依赖
  target 'YourProjectTarget' do
    # 添加 pod 'TUILiveKit' 依赖
    pod 'TUILiveKit'

  end
```

2. **安装组件：**

   在终端中 `cd` 到 `Podfile` 文件所在的目录，然后执行以下命令安装组件。

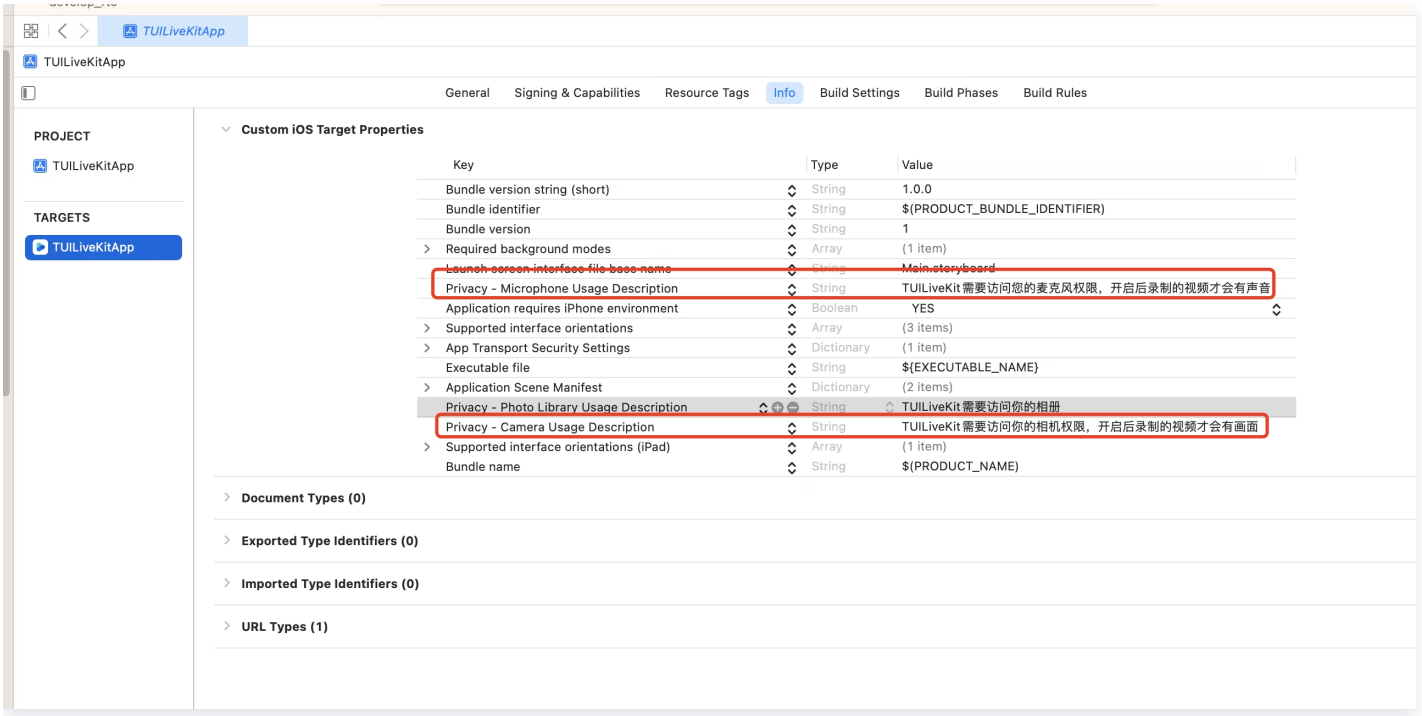
```
pod install
```

步骤 2：工程配置

为了使用音视频功能，您的应用需要获取麦克风和摄像头的权限。请在应用的 `Info.plist` 文件中添加以下两项，并填写对应的使用说明，这些说明将在系统请求权限时向用户显示：

```
<key>NSCameraUsageDescription</key>
```

```
<string>TUILiveKit需要访问你的相机权限，开启后录制的视频才会有画面</string>
<key>NSMicrophoneUsageDescription</key>
<string>TUILiveKit需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```



完成登录

代码集成完成后，您需要完成登录，这是**使用 TUILiveKit 的关键步骤**，因为只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

> ⚠ **说明：**
> 在示例代码中，登录操作是在 `didFinishLaunchingWithOptions` 方法内完成的。但在实际项目场景下，强烈推荐 **您在完成自己的用户身份验证等相关登录操作后，再调用 TUILiveKit 的登录服务**。这样可以避免因过早调用登录服务，导致业务逻辑混乱或数据不一致的问题，同时也能更好地适配您项目中现有的用户管理和权限控制体系。

Swift

```
//
//  AppDelegate.swift
//

// 1. 导入 TUICore
import TUICore
```

```
// 2. 示例代码在 didFinishLaunchingWithOptions 完成登录，推荐您在自己登录业务完成后再调用登录服务
func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {

    // 3. 调用登录接口
    TUILogin.login(1400000001,                // 请替换为开通服务控制台的 SDKAppID
            userID: "denny",                  // 请替换为您的 UserID
            userSig: "xxxxxxxxxxx") {         // 您可以在控制台中计算一个 UserSig 并填在这个位置
      print("login success")
    } fail: { (code, message) in
      print("login failed, code: \(code), error: \(message ?? "nil")")
    }

    return true
}
```

Objective-C

```
//
//  AppDelegate.m
//

// 1. 导入 TUICore
#import <TUICore/TUILogin.h>

// 2. 示例代码在 didFinishLaunchingWithOptions 完成登录，推荐您在自己登录业务完成后再调用登录服务
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    // 3. 调用登录接口
    [TUILogin login:1400000001          // 请替换为开通服务控制台的 SDKAppID
             userID:@"denny"            // 请替换为您的 UserID
            userSig:@"xxxxxxxxxxx"      // 您可以在控制台中计算一个 UserSig 并填在这个位置
```

```
                succ:^{
        NSLog(@"login success");
    } fail:^(int code, NSString * _Nullable msg) {
        NSLog(@"login failed, code: %d, error: %@", code, msg);
    }];
    return YES;
}
```

登录接口参数说明：

参数	类型	说明
SDKAppID	Int	从 控制台 获取，中国站通常是以 `140` 或 `160` 开头的 10 位整数。
userID	String	当前用户的唯一 ID，仅包含英文字母、数字、连字符和下划线。为避免多端登录冲突，**请勿使用** `1` 、`123` **等简单 ID**。
userSig	String	用于腾讯云鉴权的票据。请注意： • 开发环境：您可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 userSig 或者 通过 UserSig 辅助工具 生成临时的 UserSig。 • 生产环境：为了防止密钥泄露，请务必采用服务端生成 UserSig 的方式。详细信息请参考 服务端生成 UserSig 。 更多信息请参见 如何计算及使用 UserSig 。

登录异常状态处理【可选】

`TUILogin` 提供了登录状态回调机制，方便您处理可能出现的登录异常情况，主要包括 **“被踢下线”** 和 **“签名过期”** 这两种异常状态的回调：

- **被踢下线**：用户在线情况下被踢，IM SDK 会通过 `onKickedOffline` 回调通知您，此时可以在 UI 提示用户，并调用 `TUILogin.login` 重新登录。
- **签名过期**：用户在线期间收到 onUserSigExpired 回调，说明您之前给该用户签发的 userSig 已经过期了，这个时候如果当前用户在您后台的登录态依然有效，您可以让您的 app 向您的后台请求新的 userSig，并调用 TUILogin.login 续签登录态。

Swift

```
// YourLoginService 代表您负责登录的业务模块
class YourLoginService: NSObject {

    // 监听登录状态回调
```

```
    func addLoginListener() {
        TUILogin.add(self)
    }
    // 取消监听登录状态回调
    func removeLoginListener() {
        TUILogin.remove(self)
    }
}

// 实现登录回调 TUILoginListener
extension YourLoginService: TUILoginListener {

    // 用户被踢下线回调
    func onKickedOffline() {
      // 您的业务代码：UI 交互提示用户，然后重新登录
    }


     // 用户签名过期回调
    func onUserSigExpired() {
      // 您的业务代码：如果当前用户在您后台的登录态依然有效，您可以让您的 app 向您的后台请求新的 userSig，并调用 TUILogin.login 续签登录态。
    }
}
```

Objective-C

```
@interface YourLoginService() <TUILoginListener>

// 监听登录状态回调
- (void)addLoginListener;

// 取消监听登录状态回调
- (void)removeLoginListener;

@end

@implementation YourLoginService
```

```
// 监听登录状态回调
- (void)addLoginListener {
    [TUILogin add:self];
}

// 取消监听登录状态回调
- (void)removeLoginListener {
    [TUILogin remove:self];
}

#pragma mark - TUILoginListener

// 用户被踢下线回调
- (void)onKickedOffline {
    // 您的业务代码：UI 交互提示用户，然后重新登录
}

// 用户签名过期回调
- (void)onUserSigExpired {
    // 您的业务代码：UI 交互提示用户，然后重新登录
}

@end
```

下一步

恭喜您，现在您已经成功集成了**直播**组件并完成了登录。接下来，您可以根据您的业务场景实现**主播开播**、**观众观看**、**直播列表**等功能。

视频直播场景

功能	描述	集成指引
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
观众观看	实现观众进入主播的直播间后观看直播，实现观众连麦 、直播间信息、在线观众、弹幕显示等功能。	观众观看
直播列表	展示直播列表界面和功能，包含直播列表、房间信息展示功能。	直播列表

语聊房场景

功能	描述	集成指引
主播开播	主播创建语聊房全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
观众观看	观众进入语聊房后收听，实现上麦、弹幕显示等功能。	观众观看
直播列表	展示语聊房列表界面和功能，包含语聊房列表、房间信息展示功能。	直播列表

常见问题

pod install 执行后本地安装找不到 TUILiveKit 最新版本?

如果无法安装 TUILiveKit 最新版本，请按以下步骤操作：

1. 在 Podfile 所在目录下删除 `Podfile.lock` 和 `Pods`，您可以选择手动删除或终端执行以下命令

```
// cd 到 Podfile 所在目录下

rm -rf Pods/
rm Podfile.lock
```

2. 在 Podfile 所在目录下执行 `pod install --repo-update`

```
// cd 到 Podfile 所在目录下

pod install --repo-update
```

每次进房都需要调用登录吗?

不需要。通常您只需要完成一次 `TUILogin.login` 调用即可，我们建议您将 `TUILogin.login` 和 `TUILogin.logout` 与自己的登录业务关联。

Podfile 文件有没有示例配置可以参考?

您可以参考 GitHub TUILiveKit Example 工程 `Podfile` 示例文件。

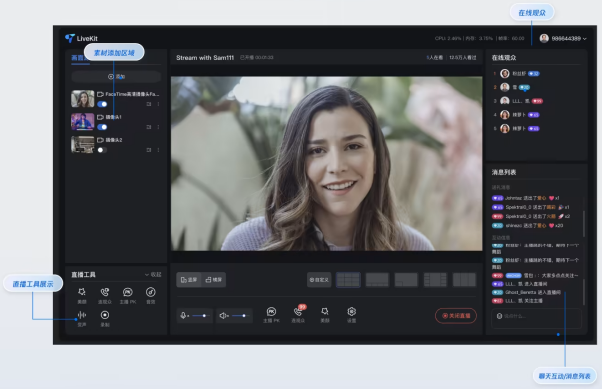
准备工作（Web Vue3）

最近更新时间：2026-03-03 18:01:03

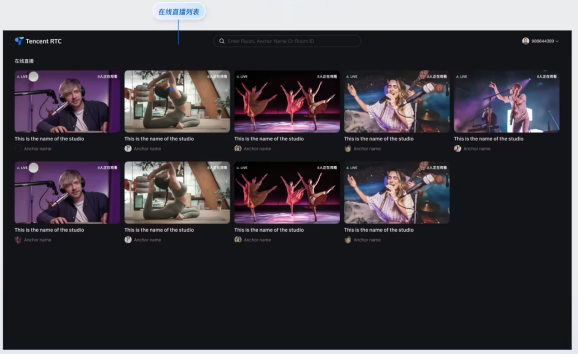
功能预览

TUILiveKit 是一个功能全面的直播组件，集成后可快速实现以下功能模块：

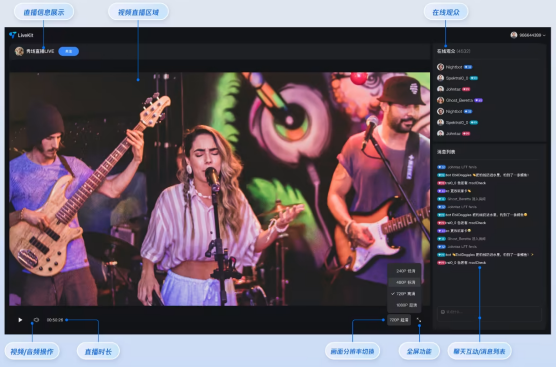
主播直播页



直播列表页



观众观看页



准备工作

1. 开通服务

在使用 TUILiveKit 前，请先参考 开通及续费 TUILiveKit，领取 TUILiveKit 体验版或者开通付费版。

2. 环境要求

- **现代浏览器:** 支持 WebRTC APIs 的现代浏览器。
- **设备:** 摄像头、麦克风、扬声器。

3. 配置要求

- **Node.js:** ≥ 18.19.1（推荐使用官方 LTS 版本，可通过 `node -v` 命令查看当前版本）
- **Vue:** ≥ 3.4.21

- **Vite:** ⩾ 5.4.0
- **TypeScript:** ⩾ 5.8.3
- **Sass:** ⩾ 1.77.0

代码集成

步骤 1：安装依赖

npm

```
npm install tuikit-atomicx-vue3 @tencentcloud/uikit-base-component-vue3
--save
```

pnpm

```
pnpm install tuikit-atomicx-vue3 @tencentcloud/uikit-base-component-vue3
```

yarn

```
yarn add tuikit-atomicx-vue3 @tencentcloud/uikit-base-component-vue3
```

步骤 2：完成登录

完成登录是使用 TUILiveKit 的基础，只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

> ⚠ **说明：**
> 在示例代码中，直接进行了登录接口的调用。但在实际项目场景下，强烈推荐**您在完成自己的用户身份验证等相关登录操作后，再调用 TUILiveKit 的登录服务**。这样可以避免因过早调用登录服务，导致业务逻辑混乱或数据不一致的问题，同时也能更好地适配您项目中现有的用户管理和权限控制体系。

```
// 强烈建议：在您的业务系统完成登录后，立刻执行以下登录逻辑
import { useLoginState } from 'tuikit-atomicx-vue3';

const { login, logout } = useLoginState();

async function initLogin() {
```

```
  try {
    await login({
      sdkAppId: 0,          // SDKAppID - 开通服务时获取的 SDKAppID
      userId: '',           // UserID   - 用户 ID
      userSig: '',          // userSig  - 用户签名，详细获取方式请参照下文参数说
明
    });
  } catch (error) {
    console.error('登录失败:', error);
  }
}
```

注意：

安全提醒：出于安全考虑，强烈建议将 `userSig` 的计算逻辑放在您的服务端进行，避免将 `SecretKey` 暴露在前端代码中。您可以使用 控制台辅助工具 生成临时 `userSig` 进行调试。更多信息参见 如何计算及使用 UserSig。GitHub 中的示例代码使用了 `genTestUserSig` 函数在本地计算 `userSig` 是为了更快地让您跑通当前的接入流程，但该方案会将您的 `SecretKey` 暴露在代码当中，这并不利于您后续升级和保护您的 `SecretKey`。参数说明如下表所示：

参数	类型	说明
SDKAppID	int	从 控制台 获取，中国站通常是以 `140` 或 `160` 开头的 10 位整数。
UserID	String	当前用户的唯一 ID，仅包含英文字母、数字、连字符和下划线。为避免多端登录冲突，**请勿使用** `1`、`123` **等简单 ID**。
userSig	String	用于腾讯云鉴权的票据。请注意： • 开发环境：您可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 userSig 或者 通过 UserSig 辅助工具 生成临时的 UserSig。 • 生产环境：为了防止密钥泄露，请务必采用服务端生成 UserSig 的方式。详细信息请参考 服务端生成 UserSig。 更多信息请参见 如何计算及使用 UserSig。

下一步

完成登录后，您就可以开始集成 TUILiveKit 的直播功能模块了，可以参照下表链接，接入主播开播、观众观看和直播列表功能页面。

功能	描述	体验链接
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播（Web Vue3）
观众观看	实现观众进入主播的直播间后观看直播，实现观众连麦 、直播间信息、在线观众、弹幕显示等功能。	观众观看（Web React） 观众观看（Web Vue3）
直播列表	展示直播列表界面和功能，包含直播列表，房间信息展示功能。	直播列表（Web React） 直播列表（Web Vue3）

常见问题

每次进房都需要调用登录吗?

不需要。通常您只需要调用一次 `useLoginState()` 返回的 `login` 函数，我们强烈推荐您将 `useLoginState()` 返回的 `login()` 和 `logout()` 响应式函数， 与您自己的登录业务关联。

为什么接入 TUILiveKit 后，必须使用 HTTPS 协议部署?

如果您需要部署项目打包的 dist 文件，生产环境下必须使用 **HTTPS 域名**。TUILiveKit 底层依赖 WebRTC 功能，也可能访问用户的摄像头、麦克风、扬声器等物理设备，浏览器厂商为了保护用户数据安全和隐私安全，要求必须使用 HTTPS 协议。

注意：

URL 域名协议限制

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

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

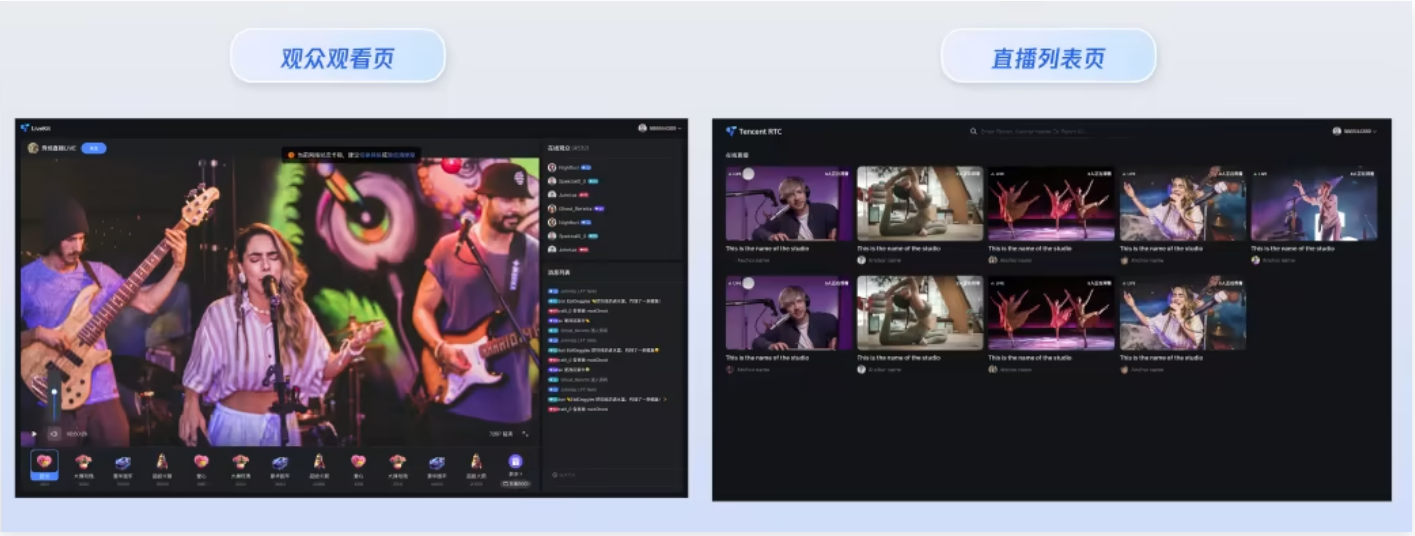
本地开发环境	http://[本机IP]	支持	不支持	不支持	–
本地开发环境	file:///	支持	支持	支持	–

准备工作（Web React）

最近更新时间：2026-03-03 18:01:03

功能预览

TUILiveKit React 是一个功能丰富的 Web 直播组件，集成后可快速实现以下功能模块：



准备工作

开通服务

在使用 TUILiveKit 前，请先参考 开通及续费 TUILiveKit，领取 TUILiveKit 体验版或者开通付费版。

环境要求

- **现代浏览器:** 支持 WebRTC APIs 的现代浏览器
- **设备：**摄像头、麦克风、扬声器

配置要求

- **Node.js:** ≥ 18.19.1（推荐使用官方 LTS 版本，可通过 `node -v` 命令查看当前版本）
- **react:** ≥ 18.2.0 且 < 19.0.0
- **vite:** ≥ 5.4.0
- **typescript：** ≥ 5.8.3
- **sass：** ≥ 1.77.0

代码集成

步骤 1：安装依赖

npm

```
npm install tuikit-atomicx-react @tencentcloud/uikit-base-component-
react --save
```

pnpm

```
pnpm install tuikit-atomicx-react @tencentcloud/uikit-base-component-
react
```

yarn

```
yarn install tuikit-atomicx-react @tencentcloud/uikit-base-component-
react
```

步骤 2：完成登录

完成登录是使用 TUILiveKit 的基础，只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

> ⚠ **说明：**
> 在示例代码中，直接进行了登录接口的调用。但在实际项目场景下，强烈推荐**您在完成自己的用户身份验证等相关登录操作后，再调用 TUILiveKit 的登录服务**。这样可以避免因过早调用登录服务，导致业务逻辑混乱或数据不一致的问题，同时也能更好地适配您项目中现有的用户管理和权限控制体系。

```
// 强烈建议：在您的业务系统完成登录后，立刻执行以下登录逻辑
import { useLoginState } from 'tuikit-atomicx-react';

const { login, logout } = useLoginState();

async function initLogin() {
  try {
    await login({
      SDKAppID: 0,        // SDKAppID - 开通服务时获取的 SDKAppID
      userID: '',         // UserID   - 用户 ID
      userSig: '',        // userSig  - 用户签名，详细获取方式请参照下文参数说
明
```

```
    });
  } catch (error) {
    console.error('登录失败:', error);
  }
}
```

注意：

安全提醒：出于安全考虑，强烈建议将 `userSig` 的计算逻辑放在您的服务端进行，避免将 `SecretKey` 暴露在前端代码中。您可以使用 控制台辅助工具 生成临时 `userSig` 进行调试。更多信息参见 如何计算及使用 UserSig。GitHub 中的示例代码使用了 `genTestUserSig` 函数在本地计算 `userSig` 是为了更快地让您跑通当前的接入流程，但该方案会将您的 `SecretKey` 暴露在代码当中，这并不利于您后续升级和保护您的 `SecretKey`。参数说明如下表所示：

参数	类型	说明
SDKAppID	int	从 控制台 获取，中国站通常是以 `140` 或 `160` 开头的 10 位整数。
UserID	String	当前用户的唯一 ID，仅包含英文字母、数字、连字符和下划线。为避免多端登录导致的账号互踢，**请勿使用 `1`、`123` 等简单 ID**。
userSig	String	用于腾讯云鉴权的票据。请注意： • 开发环境：您可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 userSig 或者 通过 UserSig 辅助工具 生成临时的 UserSig。 • 生产环境：为了防止密钥泄露，请务必采用服务端生成 UserSig 的方式。详细信息请参考 服务端生成 UserSig。 更多信息请参见 如何计算及使用 UserSig。

下一步

完成登录后，您就可以开启集成 TUILiveKit 的直播功能了，可以参照下表链接，接入：直播观看、直播列表等功能。

功能	描述	体验链接
观众观看	实现观众进入主播的直播间后观看直播，实现观众连麦 、直播间信息、在线观众、弹幕显示等功能	观众观看（Web React）

直播列表	展示直播列表界面和功能，包含直播列表，房间信息展示功能	直播列表（Web React）
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动	主播开播（Web Vue3）（暂不支持 React）

常见问题

每次进房都需要调用登录吗?

不需要。通常您只需要调用一次 useLoginState() 返回的 login 函数，我们强烈推荐您将 useLoginState() 返回的 login() 和 logout() 响应式函数， 与您自己的登录业务关联。

为什么接入 TUILiveKit 后，必须使用 HTTPS 协议部署?

如果您需要部署项目打包的 dist 文件，生产环境下必须使用 **HTTPS 域名**。TUILiveKit 底层依赖 WebRTC 功能，也可能访问用户的摄像头、麦克风、扬声器等物理设备，浏览器厂商为了包含用户数据安全和隐私安全，要求必须使用 HTTPS 协议。

注意：

页面访问协议说明

浏览器厂商出于对用户安全、隐私等问题的考虑，限制网页在 https 协议下才能正常使用 TRTC Web SDK（WebRTC）的全部功能。为确保生产环境用户顺畅接入和体验 TRTC Web SDK 的全部功能，请使用 https 协议访问音视频应用页面。

注：本地开发可以通过 http://localhost 或者 file:// 协议进行访问。

URL域名及协议支持情况请参考如下表格：

应用场景	协议	接收（拉流）	发送（推流）	屏幕分享	备注
生产环境	https协议	支持	支持	支持	推荐
生产环境	http协议	支持	不支持	不支持	
本地开发环境	http://localhost	支持	支持	支持	推荐
本地开发环境	http://127.0.0.1	支持	支持	支持	
本地开发环境	http://[本机IP]	支持	不支持	不支持	
本地开发环境	file://	支持	支持	支持	

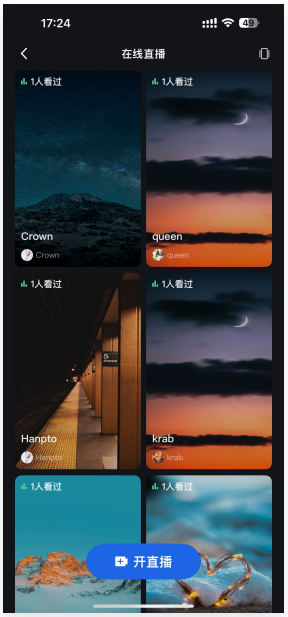
准备工作（Flutter）

最近更新时间：2026-03-03 18:01:02

功能预览

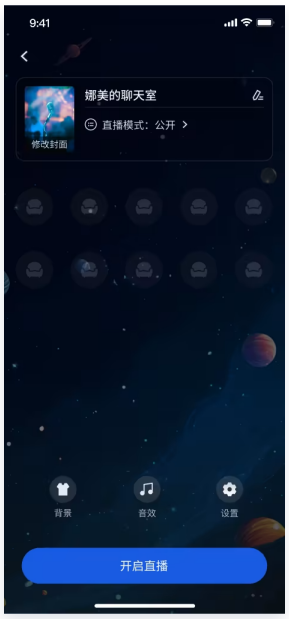
`TUILiveKit` 是一个功能全面的直播组件，集成后可快速实现以下场景功能。

视频直播

主播准备页	主播开播页	直播列表	观众观看页
			

语聊房

主播准备页	主播开播页	直播列表	观众观看页



准备工作

开通服务

在使用 TUILiveKit 前，请先参考 开通服务，领取 TUILiveKit **体验版**或开通**大规模直播版**套餐。

环境要求

- Flutter
  - Flutter 3.27.4 或更高版本。
  - Dart 3.6.2 或更高的版本。
- Android 平台
  - Android 5.0（SDK API Level 21）及以上版本。
  - Gradle 7.0 及以上的版本。
  - Android 5.0 及以上的手机设备。
- iOS 平台
  - Xcode 15 或更高版本。
  - iOS 13.0 或更高版本。
  - 已安装 CocoaPods 环境。如果您尚未安装，请 点击查看 安装步骤。

代码集成

步骤 1：下载 TUILiveKit 组件

在工程的根目录下，通过命令行执行以下命令安装组件 tencent_live_uikit 插件。

```
flutter pub add tencent_live_uikit
```

安装命令执行成功后，控制台将输出如下信息：

```
Resolving dependencies...
Downloading packages...
......
+ tencent_live_uikit x.x.x
......
Changed xx dependencies!
xx packages have newer versions incompatible with dependency
constraints.
Try `flutter pub outdated` for more information.
```

步骤 2：工程配置

Android

1. **如果需要编译运行在 Android 平台，由于 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单。**
   - **首先，需要在工程的 `android/app/build.gradle` 文件中配置并开启混淆规则：**

```
  android {
    ......
    buildTypes {
        release {
            ......
            // 配置并开启混淆规则
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}
```

   - **在 `android/app/proguard-rules.pro` 文件中添加以下代码，如果该文件不存在则新建一个文件：**

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

2. 在工程的 `android/app/build.gradle` 文件中配置开启 Multidex 支持。

```
android {
    ......
    defaultConfig {
        ......
        // 开启 Multidex 支持
        multiDexEnabled true
    }
}
```

3. （可选）如果您需要接入浮窗版直播页面，需要开启系统画中画特性。

   请在 App 主工程的 `AndroidManifest.xml` 里设置 `MainActivity` 的 `android:supportsPictureInPicture` 为 true：

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android">
    <application>
        <activity
            android:name=".MainActivity"
            android:supportsPictureInPicture="true"
        </activity>
    </application>
</manifest>
```

iOS

1. iOS 工程编译 release 版本时，需要配置符号保留规则。使用 Xcode 打开工程，在 TARGETS 列表中选择 target（通常是 Runner），选择**项目 > Build Settings > Deployment**，将其下的 **Strip Style** 设置为 `Non-Global Symbols` ，以保留所需要的全局符号信息。此配置是必需的，否则可能在运行时出现异常，无法进入房间。
2. **（可选）**如果需要在 iOS 模拟器上调试，并且您使用的 Mac 电脑使用的是 Intel 芯片，请在工程的 `ios/Podfile` 文件中添加以下代码：

```
target 'xxxx' do
  ......
end
......

post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
      config.build_settings['VALID_ARCHS'] = 'arm64 arm64e x86_64'
      config.build_settings['VALID_ARCHS[sdk=iphonesimulator*]'] =
'x86_64'
    end
  end
end
```

3. 由于 TUILiveKit 会使用 iOS 的音视频功能，需要授权麦克风和摄像头的使用权限。

   授权操作方法：在 iOS 工程的 `Info.plist` 的第一级 `<dict>` 目录下添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

```
<key>NSCameraUsageDescription</key>
<string>CallingApp需要访问您的相机权限，开启后录制的视频才会有画面
</string>
<key>NSMicrophoneUsageDescription</key>
<string>CallingApp需要访问您的麦克风权限，开启后录制的视频才会有声音
</string>
```

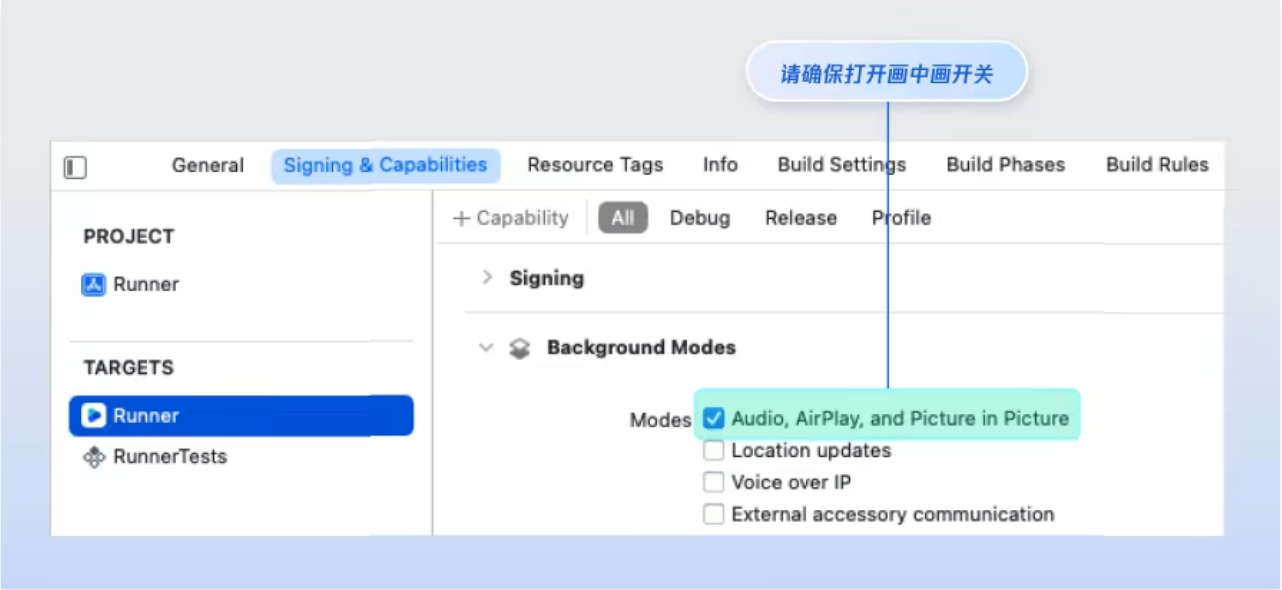
完成以上添加后，在 `ios/Podfile` 中添加以下预处理器定义，用于启用相机与麦克风权限。

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
      target.build_configurations.each do |config|
        config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||=
[
      '$(inherited)',
      'PERMISSION_MICROPHONE=1',
      'PERMISSION_CAMERA=1',
```

```
        ]
      end
    end
  end
```

4. （可选）如果您需要接入浮窗版直播页面，需要开启系统画中画特性。

   请在 Xcode 工程配置里，开启画中画：



步骤 3：配置页面导航和多语言功能

为了确保 TUILiveKit 能够正常管理页面导航和显示多语言界面，需要在 Flutter 应用框架中进行以下配置：

- 在 `navigatorObservers` 中添加 `TUILiveKitNavigatorObserver.instance`，用于监听页面路由变化和管理组件生命周期。
- 在 `localizationsDelegates` 中添加相关本地化代理，确保界面文案能够根据系统语言正确显示。

以 MaterialApp 框架为例，示例代码如下：

```
import 'package:tencent_live_uikit/tencent_live_uikit.dart';

// 您自己的APP主类
class XXX extends StatelessWidget {
  const XXX({super.key});

 @override
  Widget build(BuildContext context) {
    return MaterialApp(
```

```
      // 添加 TUILiveKit 导航观察者，用于监听页面路由变化和生命周期管理
      navigatorObservers: [TUILiveKitNavigatorObserver.instance],
      // 添加本地化代理，支持多语言文案显示
      localizationsDelegates: [
      ...LiveKitLocalizations.localizationsDelegates,
      ...BarrageLocalizations.localizationsDelegates,
      ...GiftLocalizations.localizationsDelegates],

      // 您APP的其他配置
      ......
    );
  }
}
```

配置后，组件将支持多语言并能正确处理页面跳转。

完成登录

代码集成完成后，您需要调用 `TUILogin.login` 完成登录，这是**使用 TUILiveKit 的关键步骤**，因为只有在登录成功后才能正常使用 TUILiveKit 的各项功能，请确保相关参数配置正确：

> ⚠ **说明：**
> 在实际项目场景下，强烈推荐**您在完成自己的用户身份验证等相关登录操作后，再调用 TUILiveKit 的登录服务**。这样可以避免因过早调用登录服务，导致业务逻辑混乱或数据不一致的问题，同时也能更好地适配您项目中现有的用户管理和权限控制体系。

```
import 'package:tencent_cloud_uikit_core/tencent_cloud_uikit_core.dart';
......

login() async {
  await TUILogin.instance.login(
    1400000001,      // 请替换为开通服务控制台的 SDKAppID
    "denny",         // 请替换为您的 UserID
    "xxxxxxxxxxx",   // 您可以在控制台中计算一个 UserSig 并填在这个位置
    TUICallback(
      onError: (code, message) {
        print("TUILogin login fail, {code:$code, message:$message}");
      },
      onSuccess: () async {
        print("TUILogin login success");
```

```
        },
      ),
    );
}
```

登录接口参数说明：

参数	类型	说明
SDKAppID	Int	从 控制台 获取，中国站通常是以 `140` 或 `160` 开头的 10 位整数。
userID	String	当前用户的唯一 ID，仅包含英文字母、数字、连字符和下划线。为避免多端登录冲突，**请勿使用** `1`、`123` **等简单 ID**。
userSig	String	用于腾讯云鉴权的票据。请注意： • 开发环境：您可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 userSig 或者 通过 UserSig 辅助工具 生成临时的 UserSig。 • 生产环境：为了防止密钥泄露，请务必采用服务端生成 UserSig 的方式。详细信息请参考 服务端生成 UserSig 。 更多信息请参见 如何计算及使用 UserSig 。

（可选）浮窗版直播间配置

默认 `App` 的 `MaterialApp` 首页是在 `rootNavigator` 上显示。如果您希望直播间支持浮窗功能（即在直播过程中可以最小化直播窗口，悬浮在其他页面上方继续观看），需要在 `rootNavigator` 上添加一层二级导航器 `secondaryNavigator`，把首页移到 `secondaryNavigator` 上。直播页面的 `Overlay` 将会放在 `secondaryNavigator` 上展示。`secondaryNavigator` 用于承载 Overlay，并常驻 App，以实现悬浮窗的效果。

```
import 'package:tencent_live_uikit/tencent_live_uikit.dart';
import 'package:tencent_live_uikit/common/widget/global.dart';

// 您自己的 App 主类
class XXX extends StatelessWidget {
  const XXX({super.key});

 @override
  Widget build(BuildContext context) {
    return MaterialApp(
          // 首页配置
          home: Navigator(
```

```
            // Global.secondaryNavigatorKey 是 TUILiveKit 提供的用于管理浮窗
页面的全局导航器 Key。
            key: Global.secondaryNavigatorKey,
            onGenerateRoute: (settings) => MaterialPageRoute(
              settings: const RouteSettings(name: 'home_widget'),
              builder: (BuildContext context) {
                // HomeWidget 是您自己的应用首页 Widget，请替换为您实际的首页类
                return const HomeWidget();
              },
            ),
          ),
          // 您 App 的其他配置
          ......
    );
  }
}
```

此步骤是进入浮窗版页面的必要配置，完整流程请参考 添加浮窗版主播页面 和 添加浮窗版观众观看页面 。如果您不需要接入浮窗版页面，请跳过此步骤。

下一步

恭喜您，现在您已经成功集成了**直播**组件并完成了登录。接下来，您可以根据您的业务场景实现**主播开播**、**观众观看**、**直播列表**等功能。

视频直播

功能	描述	集成指引
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
观众观看	实现观众进入主播的直播间后观看直播，实现观众连麦 、直播间信息、在线观众、弹幕显示等功能。	观众观看
直播列表	展示直播列表界面和功能，包含直播列表、房间信息展示功能。	直播列表

语聊房

功能	描述	集成指引
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播

观众观看	实现观众进入主播的直播间后观看直播，实现观众连麦 、直播间信息、在线观众、弹幕显示等功能。	观众观看
直播列表	展示直播列表界面和功能，包含直播列表、房间信息展示功能。	直播列表

常见问题

关于每次进房是否需要登录?

不需要。通常只需要完成一次 `TUILogin.login` 调用即可，我们建议您将 `TUILogin.login` 和 `TUILogin.logout` 与自己的登录业务关联。

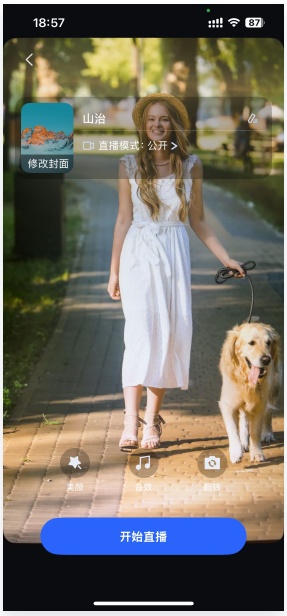
iOS release下运行或打包后无法进入房间?

请参考 工程配置 中 iOS 部分第 1 步：在 Xcode 中，在 TARGETS 列表中选择 target（通常是 Runner），选择**项目** > **Build Settings** > **Deployment**，将其下的 **Strip Style** 设置为 `Non-Global Symbols` ，以保留所需要的全局符号信息，否则可能在运行时出现异常，无法进入房间。

准备工作（uni-app 客户端）

最近更新时间：2026-03-04 17:57:42

功能预览

主播预览页	主播直播页	连麦管理页	观众观看页
			

准备工作

开通服务

在使用 TUILiveKit 前，您需要在 TRTC 控制台 开通相关服务，并领取体验版或者开通付费版。

开发环境要求

> **说明：**
> HBuilderX 4.64 和 4.65 版本对 UTS 插件打包存在已知兼容性问题，建议使用 4.66 或更高版本。

- HBuilderX：HBuilderX 是 **uni-app 官方**的集成开发环境（IDE），我们需要用它来导入、配置和运行我们的 Demo。
- 2个移动设备： Android 5.0 以上的设备 / iOS 13.0 及以上设备。

代码集成

步骤 1：下载 TUILiveKit 源码

1. 克隆项目源码

```
git clone https://github.com/Tencent-RTC/TUIKit_uni-app.git
```

2. 导入 atomic-x 组件

- 将源码中 `TUIKit_uni-app/App/uni_modules` 目录下的 `tuikit-atomic-x` 文件夹，完整拷贝到您项目根目录下的 `uni_modules` 文件夹中。
- 将源码中 `TUIKit_uni-app/App/components` 下的文件，完整的拷贝到您项目的 `components` 文件夹中。
- 将源码中 `TUIKit_uni-app/App/static` 目录下的文件，完整拷贝到您项目根目录下的同名文件夹下。

3. 拷贝直播开播等页面

- 将源码中 `TUIKit_uni-app/App/pages` 目录下的文件，完整拷贝到您项目根目录下的同名文件夹中。
- 将源码中 `TUIKit_uni-app/App/debug` 文件夹拷贝到您项目的根目录下，这个目录提供有可以方便您在客户端快速生成的 UserSig 的快捷函数。

4. 合并 App.vue 配置。

TUIKit_uni-app/App/App.vue 文件中包含直播能力的初始化逻辑，因此您需要将 TUIKit_uni-app/App/App.vue 文件和您项目下的 App.vue 文件进行合并。

> ⚠ **注意：**
> 不要直接覆盖 App.vue 文件！ 打开源码中的 TUIKit_uni-app/App/App.vue， 将其 `<script>` 部分的 ts 内容追加到您自己项目的 App.vue 的 `<script>` 标签内。

步骤 2：工程配置

1. 配置 `manifest.json`

   打开您项目的 manifest.json 文件，在 app-plus > distribute 节点下，确保添加了以下必要的权限：

   - Android 平台（ `android` 节点）：在 `permissions` 数组中，请确保包含以下权限：

```
"<uses-permission android:name=\"android.permission.CAMERA\"/>",
"<uses-permission android:name=\"android.permission.RECORD_AUDIO\"
/>",
"<uses-permission android:name=\"android.permission.INTERNET\"/>",
"<uses-permission
android:name=\"android.permission.ACCESS_NETWORK_STATE\"/>",
"<uses-permission
android:name=\"android.permission.WRITE_EXTERNAL_STORAGE\"/>"
```

- iOS 平台（ios 节点）：在 `privacyDescription` 对象中，请确保包含以下描述：

```
"NSCameraUsageDescription" : "应用需要访问您的相机以进行直播",
"NSMicrophoneUsageDescription" : "应用需要访问您的麦克风以进行直播"
```

在 `UIBackgroundModes` 数组中，添加 `audio` 以支持后台播放音频。

```
"UIBackgroundModes" : [ "audio" ]
```

2. 制作自定义基座。

> **说明：**
> 由于 TUILiveKit 包含原生代码（UTS 插件），您**必须**制作自定义调试基座才能在真机上运行。

2.1 在 HBuilderX 顶部菜单选择**运行 > 运行到手机或模拟器 > 制作自定义调试基座**。

2.2 制作成功后，再次选择**运行 > 运行到手机或模拟器**，在弹窗中勾选**使用自定义调试基座**后运行到您的手机。

具体操作指引参见 跑通 Demo 的 步骤3 。

完成登录

集成完成后，您需要完成登录。在您项目中需要使用到直播能力的地方进行登录，这是**使用 TUILiveKit 的关键步骤**，因为只有在登录成功后才能正常使用 TUILiveKit 的各项功能，故请您耐心检查相关参数是否配置正确：

```
import { useLoginState } from "@/uni_modules/tuikit-atomic-
x/state/LoginState";

const { login } = useLoginState();

onMounted(() => {
  login({
    sdkAppID: 1400000001,      // 请替换为开通服务控制台的 SDKAppID
    userID: "denny",          // 请替换为您的 UserID
    userSig: "xxxxxxxxxxx",   // 您可以在控制台中计算一个 UserSig 并填在这个位置
  });
});
```

登录接口参数说明

参数	类型	说明
SDKAppID	Number	从 TRTC 控制台 > 应用管理 获取。
userID	String	当前用户的唯一 ID，仅包含英文字母、数字、连字符和下划线。
userSig	String	用于腾讯云鉴权的票据。请注意： • 开发环境：您可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 UserSig 或者通过 UserSig 辅助工具 生成临时的 UserSig。 • 生产环境：为了防止密钥泄露，请务必采用服务端生成 UserSig 的方式。详细信息请参见 服务端生成 UserSig 。 更多信息请参见 如何计算及使用 UserSig 。

接入直播功能

恭喜您，现在您已经成功集成了视频直播组件并完成了登录。接下来，您可以开始接入 主播开播 、观众观看 功能或实现其他直播功能。