

多人音视频房间 SDK

快速接入



腾讯云

【 版权声明 】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【 商标声明 】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【 服务声明 】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【 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

快速接入

Web&H5 (Vue2/Vue3)

Android

iOS

uni-app (小程序)

uni-app (客户端)

Flutter

Electron (Windows&Mac)

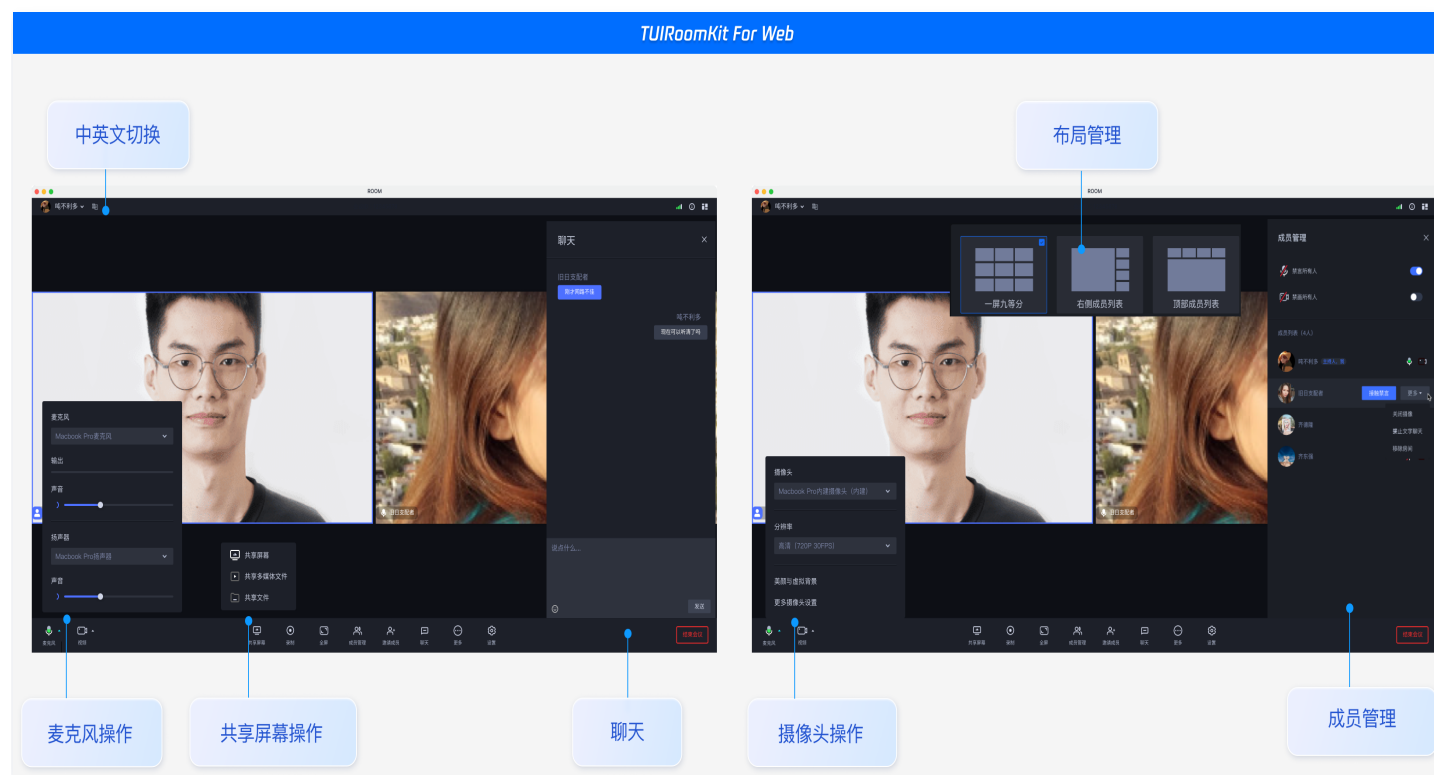
快速接入

Web&H5 (Vue2/Vue3)

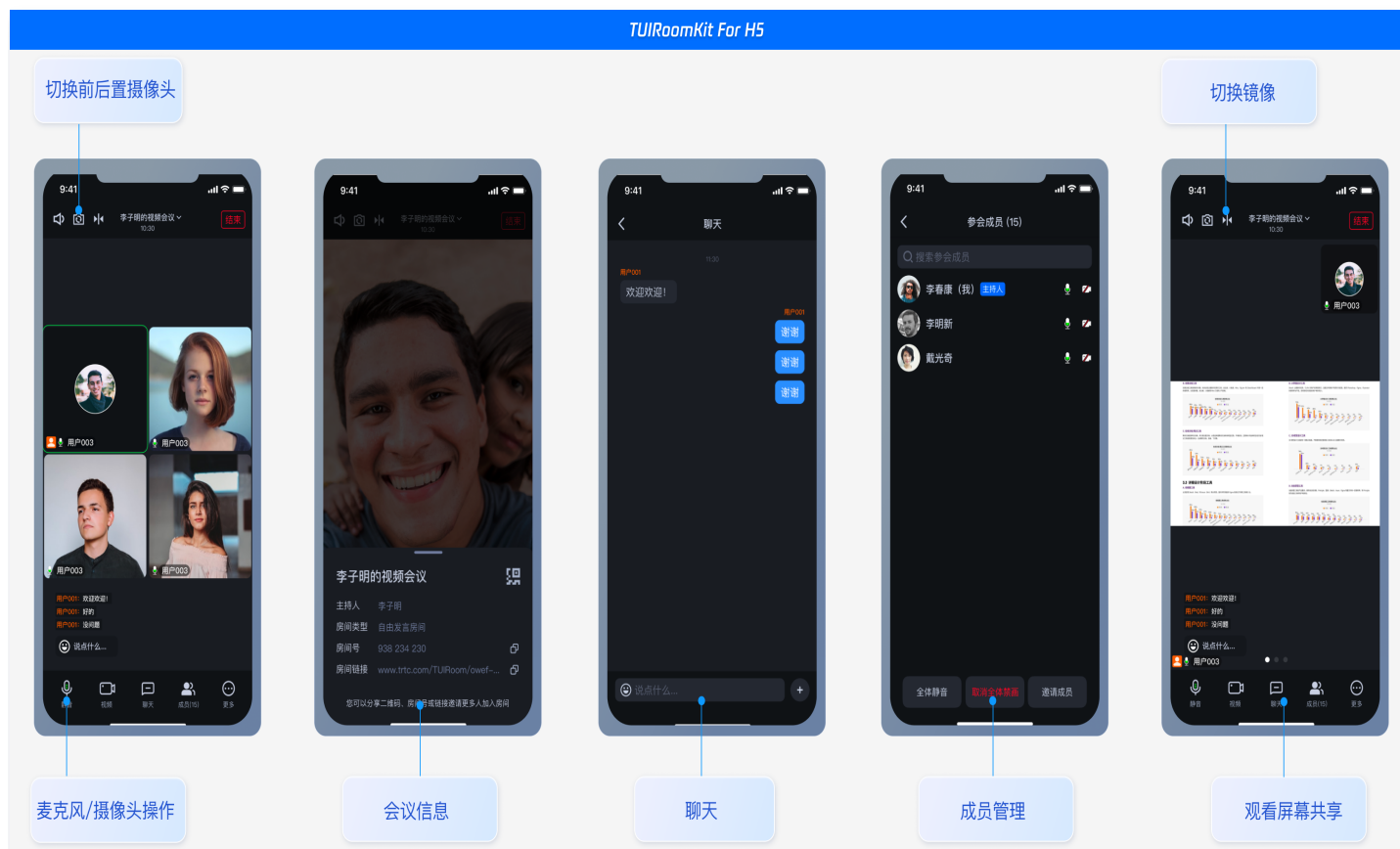
最近更新时间：2024-08-09 18:06:52

TUIRoomKit 是腾讯云推出的多人音视频 UI 组件。组件提供房间管理，音视频控制，屏幕共享，成员管理，麦位管理，即时聊天，自定义布局切换等丰富的功能交互，同时支持中英文切换，一键换肤等能力。

本文介绍 TUIRoomKit (Web & H5) 的接入指引，助力您快速上线企业会议、在线教育、医疗问诊、在线巡视、远程定损等业务场景。



TUIRoomKit For H5



TUIRoomKit Demo 体验

您可以单击 [TUIRoomKit 在线体验链接](#) 体验 TUIRoomKit 更多功能。

您可以单击 [Github](#) 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Web 示例工程。

环境准备

- Node.js 版本: Node.js $\geq$ 16.19.1 (推荐使用官方 LTS 版本, npm 版本请与 node 版本匹配)。
- Modern browser, [supporting WebRTC APIs](#).

集成 TUIRoomKit 组件

❗ 说明:

如果您当前尚无 Vue 项目, 您可以直接参见 [跑通 Demo](#) 跑通 [Github](#) 示例项目。

如果需要在已有项目中集成, 请按照以下步骤进行接入。

步骤一: 安装依赖

Vue3

```
npm install @tencentcloud/roomkit-web-vue3 pinia --save
```

Vue2

```
# 注意此处要求 vue版本 >= 2.7.16，如安装失败请检查您的 vue 版本是否支持
npm install @tencentcloud/roomkit-web-vue2.7 pinia
```

❗ 说明：

TUIRoomKit npm 包提供了会前预览组件、会中组件以及发起会议、加入会议和界面微调的方法。如需了解更多，请参见 [TUIRoomKit API](#)。若这些 API 无法满足您的业务需求，您可以参见 [UIKit 源码导出](#) 方案接入 TUIRoomKit 源码。

步骤二：项目工程配置

注册pinia： TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia。项目入口文件为 `src/main.ts` 文件。

Vue3

```
// src/main.ts 文件
import { createPinia } from 'pinia';

const app = createApp(App);
// 注册 pinia
app.use(createPinia());
app.mount('#app')
```

Vue2

```
// src/main.ts 文件
import { createPinia, PiniaVuePlugin } from 'pinia';

Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
```

```
pinia,  
render: h => h(App),  
}).$mount('#app');
```

步骤三：引用 TUIRoomKit 组件

❗ 说明：

引入 ConferenceMainView 组件，组件默认处于 **常驻模式**（组件始终显示，内部不控制组件的显示与隐藏，若业务端不进行控制，则组件一直保持显示状态）。

Vue3

```
<template>  
  <ConferenceMainView></ConferenceMainView>  
</template>  
<script setup>  
import { ConferenceMainView } from '@tencentcloud/roomkit-web-vue3';  
</script>
```

Vue2

```
<template>  
  <ConferenceMainView></ConferenceMainView>  
</template>  
<script>  
import { ConferenceMainView } from '@tencentcloud/roomkit-web-vue2.7';  
export default {  
  components: {  
    ConferenceMainView,  
  },  
};  
</script>
```

步骤四：登录 TUIRoomKit 组件

开启会议前需要调用 **login** 接口进行登录。获取 sdkAppId、userId、userSig 可参见 [开通服务](#)。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
```

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.login({
  // 获取 sdkAppId 可参考文档开通服务部分，
  https://cloud.tencent.com/document/product/1690/89140
  sdkAppId: 0,
  // 用户在您业务中的唯一标示 Id
  userId: '',
  // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool
  // 页面快速生成 userSig，注意 userSig 与 userId 为一一对应关系
  userSig: '',
});
```

参数说明

这里详细介绍一下 `login` 函数中所需要用到的几个关键参数：

- **sdkAppId**：在 [开通服务](#) 中的最后一步中获取。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **userSig**：通过对在 [开通服务](#) 中获取的 `SDKSecretKey` 对 `SDKAppID`、`UserID` 等信息进行加密，就可以得到 `UserSig`，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 `TRTC` 的服务。您可以通过控制台中的 [辅助工具](#) 生成一个临时可用的 `UserSig`。
- 更多信息请参见 [如何计算及使用 UserSig](#)。

⚠ 注意：

这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：

- **sdkAppId** 设置错误，国内站的 `SDKAppID` 一般是以140开头的10位整数。
- **UserSig** 被错配成了加密密钥（`SecretKey`），`UserSig` 是用 `SecretKey` 把 `SDKAppID`、`UserID` 以及过期时间等信息加密得来的，而不是直接把 `SecretKey` 配置成 `UserSig`。
- **userId** 被设置成“1”、“123”、“111”等简单字符串，由于 `TRTC` 不支持同一个 `UserID` 多端登录，所以在多人协作开发时，形如“1”、“123”、“111”这样的 `UserID` 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 `UserID`。

[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`，所以我们强烈建议您将 `UserSig` 的计算逻辑放在服务端进行，并在每次使用 `TUIRoomKit` 组件时向您的服务器请求实时计算出的 `UserSig`。

步骤五：发起新的会议

会议主持人可以通过调用 `start` 接口来发起一场新的会议，其他参会议者可以参见 [步骤六](#) 的描述，调用 `join` 接口加入该会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
const startConference = async () => {
  await conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，
    https://cloud.tencent.com/document/product/1690/89140
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
    userId: '',
    // 本地开发调试可在
    https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig，注意 userSig 与 userId 为一一对应关系
    userSig: '',
  });
  await conference.start('123456', {
    roomName: 'TestRoom',
    isSeatEnabled: false,
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
}
startConference()
```

步骤六：进入已有会议

参与者可以通过调用 `join` 接口，填写对应的 `roomId` 参数，来加入由会议主持人在 [步骤五](#) 中发起的会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
const joinConference = async () => {
  await conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，
    https://write.woa.com/document/139678220708134912#d9d2b8d1-f19f-441b-a436-457748c94997
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
    userId: '',
    // 本地开发调试可在
    https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig，注意 userSig 与 userId 为一一对应关系
  });
}
```

```
    userSig: '',
  });
  await conference.join('123456', {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
}
joinConference()
```

开发环境运行

1. 执行开发环境命令。（此处以 vue3 + vite 默认项目为例，不同项目 dev 指令可能不同，请根据您的项目进行调整）

```
npm run dev
```

2. 根据控制台提示，在浏览器中打开页面，如：`http://localhost:3000/`。
3. 体验 TUIRoomKit 组件功能。

生产环境部署

1. 打包 dist 文件。

```
npm run build
```

2. 部署 dist 文件到服务器上。

⚠ 注意：

生产环境要求使用 HTTPS 域名，参见下图：

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	

其他文档

- [TUIRoomKit](#)
- [TUIRoom Demo 快速跑通](#)
- [设置昵称、头像](#)
- [预定会议](#)
- [界面定制 \(TUIRoomKit\)](#)
- [常见问题](#)

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 [腾讯云实时音视频 TRTC 技术交流群](#) 进行技术交流和问题反馈。

Android

最近更新时间：2024-08-09 18:06:52

本文将指导您如何在较短时间内接入 `TUIRoomKit` 组件。遵循本指南，您将在10分钟内完成以下关键步骤，并最终实现具备完整 UI 界面的音视频会议功能。

会议界面及部分功能效果展示



环境准备

- 最低兼容 Android 4.4（SDK API Level 19），建议使用 Android 5.0（SDK API Level 21）及以上版本。
- Android Studio 3.5 及以上的版本（Gradle 3.5.4 及以上的版本）。

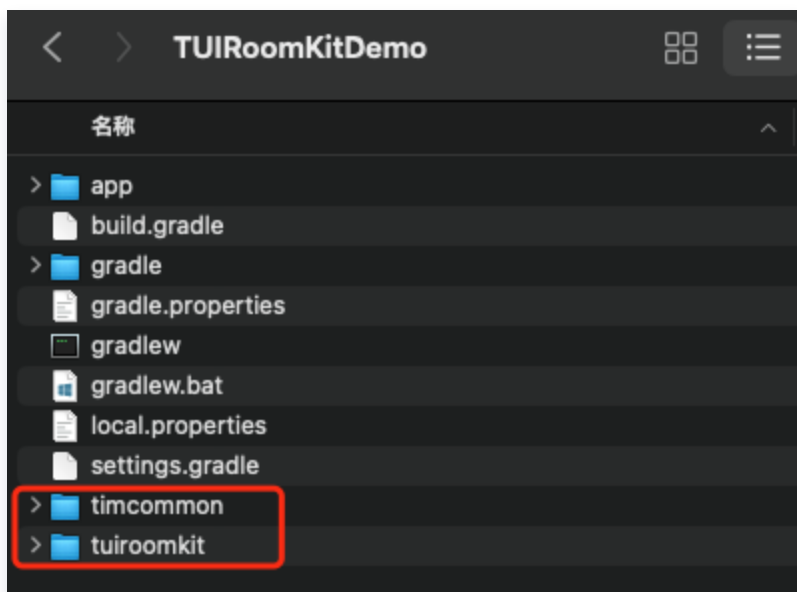
- Android 4.4 及以上的手机设备。

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要前往控制台开通 TUIRoomKit 专属的多人音视频互动服务，具体步骤请参见 [开通服务](#)。

步骤二：下载 TUIRoomKit 组件

1. 在 [Github](#) 中克隆/下载代码，然后拷贝 `Android` 目录下的 `timcommon`、`tuiroomkit` 子目录到您当前工程中的 `app` 同级目录中，如下图：



步骤三：工程配置

1. 工程根目录下找到 `setting.gradle`（或 `settings.gradle.kts`）文件，并在其中增加如下代码，它的作用是将 `tuiroomkit` 组件导入到您当前的项目中。

settings.gradle

```
include ':timcommon'
include ':tuiroomkit'
```

setting.gradle.kts

```
include(":timcommon")
include(":tuiroomkit")
```

- 在 `app` 目录下找到 `build.gradle` (或 `build.gradle.kts`) 文件，并在其中增加如下代码，它的作用是声明当前 `app` 对新加入的 `tuiroomkit` 组件的依赖。

build.gradle

```
api project(':tuiroomkit')
```

build.gradle.kts

```
api(project(":tuiroomkit"))
```

- 由于我们在 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，因此需要您在 `proguard-rules.pro` 文件中添加如下代码：

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

- 在 `app` 目录下找到 `AndroidManifest.xml` 文件，在 `application` 节点中添加 `tools:replace="android:allowBackup"`，覆盖组件内的设置，使用自己的设置。

```
// app/src/main/AndroidManifest.xml
<application
    android:name=".DemoApplication"
    android:allowBackup="false"
    android:icon="@drawable/app_ic_launcher"
    android:label="@string/app_name"
    android:largeHeap="true"
    android:theme="@style/AppTheme"
    tools:replace="android:allowBackup">
```

步骤四：登录

在您的项目中添加如下代码，它的作用是通过调用 `TUILogin` 中的相关接口完成组件的登录。这个步骤异常关键，因为只有登录后才能正常使用 `TUIRoomKit` 的各项功能，故请您耐心检查相关参数是否配置正确。

Java

```
import com.tencent.qcloud.tuicore.TUILogin;
import com.tencent.qcloud.tuicore.interfaces.TUICallback;
import com.tencent.cloud.tuikit.roomkit.debug.GenerateTestUserSig;
```

```
String userId = "denny"; // 请替换为您的 UserID
int sdkAppId = 1400000001; // 请替换为步骤一取到的 sdkAppId
String sdkSecretKey = "xxxx"; // 请替换为您的 sdkSecretKey
String userSig = GenerateTestUserSig.genTestUserSig(sdkAppId, userId,
sdkSecretKey);

TUILogin.login(context,
    sdkAppId,
    userId,
    userSig,
    new TUICallback() {
        @Override
        public void onSuccess() {
        }

        @Override
        public void onError(int errorCode, String errorMessage) {
        }
    });
```

Kotlin

```
import com.tencent.qcloud.tuicore.TUILogin
import com.tencent.qcloud.tuicore.interfaces.TUICallback
import com.tencent.cloud.tuikit.roomkit.debug.GenerateTestUserSig

val userId = "denny" // 请替换为您的 UserID
val sdkAppId = 1400000001 // 请替换为步骤一取到的 sdkAppId
val sdkSecretKey = "xxxx" // 请替换为您的 sdkSecretKey
val userSig = GenerateTestUserSig.genTestUserSig(sdkAppId, userId,
sdkSecretKey)

TUILogin.login(this,
    sdkAppId,
    userId,
    userSig,
    object : TUICallback() {
        override fun onSuccess() {
        }

        override fun onError(errorCode: Int, errorMessage: String) {
        }
    })
```

```
}  
})
```

TUILogin.login 函数参数说明:

- **SDKAppID**: 在 [开通服务](#) 中的最后一步中获取。
- **UserID**: 当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**: 使用 [开通服务](#) 的第3步中获取的 `SDKSecretKey` 对 `SDKAppID`、`UserID` 等信息进行加密，就可以得到 `UserSig`，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 `TRTC` 的服务。您可以通过控制台中的 [辅助工具](#) 生成一个临时可用的 `UserSig`。
- 更多信息请参见 [如何计算及使用 UserSig](#)。

❗ 说明:

- **开发环境**: 如果您正在本地开发调试阶段，可以采用本地 `GenerateTestUserSig.genTestUserSig()` 函数生成 `userSig`。该方法中 `SDKSecretKey` 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量。
- **生产环境**: 如果您的项目要发布上线，请采用 [服务端生成 UserSig](#) 的方式。

步骤五：发起您的第一次会议

在 [TUILogin.login](#) 成功后，参考如下代码，即可发起会议。

Java

```
// 请替换 "123456" 为您自定义的房间号  
ConferenceDefine.StartConferenceParams params = new  
ConferenceDefine.StartConferenceParams("123456");  
Intent intent = new Intent(this, ConferenceMainActivity.class);  
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);  
startActivity(intent);
```

Kotlin

```
// 请替换 "123456" 为您自定义的房间号  
val params = ConferenceDefine.StartConferenceParams("123456")  
val intent = Intent(this, ConferenceMainActivity::class.java)
```

```
intent.putExtra(KEY_START_CONFERENCE_PARAMS, params);
startActivity(intent)
```

步骤六：加入会议

在 [TUILogin.login](#) 成功后，参考如下代码，即可加入会议。

Java

```
// 请替换 "123456" 为您自定义的房间号
ConferenceDefine.JoinConferenceParams params = new
ConferenceDefine.JoinConferenceParams("123456");
Intent intent = new Intent(this, ConferenceMainActivity.class);
intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
startActivity(intent);
```

Kotlin

```
// 请替换 "123456" 为您自定义的房间号
val params = ConferenceDefine.JoinConferenceParams("123456")
val intent = Intent(this, ConferenceMainActivity::class.java)
intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
startActivity(intent)
```

界面展示

当您成功完成步骤1 – 步骤6，UI 界面效果如下：

会议主界面	用户列表
-------	------



常见问题

如果您的接入和使用中遇到问题，请参见 [常见问题](#)。

更多特性

- [设置昵称、头像](#)
- [预定会议](#)
- [会中聊天](#)

交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 [zhiliao](#)，进行技术交流和问题反馈。

iOS

最近更新时间：2024-08-29 17:20:11

本文将介绍如何快速完成 `TUIRoomKit` 组件的接入，跟随本文档，您将在一小时的时间内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的音视频会议功能。

环境准备

- iOS 13.0 及更高。
- Xcode 12.0 及更高。
- Swift 4.2 及更高。

步骤一：开通服务

在使用 `TUIRoomKit` 发起会议前，您需要前往控制台开通 `TUIRoomKit` 专属的多人音视频互动服务，具体步骤请参见 [开通服务](#)。

步骤二：导入TUIRoomKit组件

1. 在您的 `Podfile` 文件中添加以下依赖。

```
pod 'TUIRoomKit'
```

2. 执行以下命令，安装组件。

```
pod install
```

❗ 说明：

如果无法安装 `TUIRoomKit` 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表：

```
pod repo update
```

步骤三：工程配置

使用音视频功能，需要授权麦克风、摄像头和相册的使用权限。在 App 的 `Info.plist` 中添加以下几项，分别对应麦克风、摄像头和相册在系统弹出授权对话框时的提示信息。

```
<key>NSCameraUsageDescription</key>
<string>TUIRoomKit需要访问您的相机权限</string>
```

```
<key>NSMicrophoneUsageDescription</key>
<string>TUIRoomKit需要访问您的麦克风权限</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>TUIRoomKit需要访问您的相册</string>
```

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

PROJECT
DemoApp

TARGETS
DemoApp
TXReplayKit_Screen

Custom iOS Target Properties

Key	Type	Value
Bundle name	String	\$(PRODUCT_NAME)
Launch screen interface file base name	String	LaunchScreen
Application supports iTunes file sharing	Boolean	YES
Default localization	String	\$(DEVELOPMENT_LANGUAGE)
Bundle version	String	1
Required background modes	Array	(1 item)
Privacy - Camera Usage Description	String	TUIRoomKit需要访问你的相机权限
Appearance	String	
Application supports indirect input events	Boolean	YES
Privacy - Microphone Usage Description	String	TUIRoomKit需要访问您的麦克风权限
Bundle OS Type code	String	\$(PRODUCT_BUNDLE_PACKAGE_TYPE)
Bundle version string (short)	String	1.0
InfoDictionary version	String	6.0
Required device capabilities	Array	(1 item)
Executable file	String	\$(EXECUTABLE_NAME)
Supported interface orientations (iPad)	Array	(4 items)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
Application Scene Manifest	Dictionary	(2 items)
Application requires iPhone environment	Boolean	YES
Supported interface orientations	Array	(1 item)
Privacy - Photo Library Usage Description	String	TUIRoomKit需要访问你的相册

Document Types (0)

Exported Type Identifiers (0)

步骤四：登录

在您的项目中添加如下代码，它的作用是通过调用 TUICore 中的相关接口完成TUI组件的初始化。这个步骤非常关键，因为只有在登录成功后才能正常使用 TUIRoomKit 的各项功能，故请您耐心检查相关参数是否配置正确：

Swift

```
import TUICore

TUILogin.login(1400000001, // 请替换为步骤一取到的
SDKAppID
```



```
        userID: "998", // 请替换为您的 UserID
        userSig: "xxxxxxxxxx") { // 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")")
}
```

OC

```
#import "TUICore/TUILogin.h"

[TUILogin login:1400000001 // 请替换为步骤一取到的
SDKAppID
        userID:@"998" // 请替换为您的 UserID
        userSig:@"xxxxxxxxxx" succ:^(
UserSig 并填在这个位置
    NSLog(@"login, success");
} fail:^(int code, NSString * _Nullable msg) {
    NSLog(@"login, failed, code:%d, msg:%@", code, msg);
}];
```

参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- **SDKAppID**：在 [开通服务](#) 中的第3步中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**：使用 [开通服务](#) 的第3步中获取的 SDKSecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 [辅助工具](#) 生成一个临时可用的 UserSig。

更多信息请参见 [如何计算及使用 UserSig](#)。

步骤五：主持人发起快速会议

会议主界面为 ConferenceMainViewController，您只需要在登录完成后，按照如下示例创建并跳转 ConferenceMainViewController 即可发起快速会议。

Swift

```
import TUIRoomKit
```

```
// CreateConferenceViewController 为您自己的 ViewController
class CreateConferenceViewController: UIViewController {
    func quickStartConferenceAction() {
        let params = StartConferenceParams(roomId: "123456") // 请替换
        "123456" 为您自定义的会议号
        let conferenceViewController = ConferenceMainViewController()
        conferenceViewController.setStartConferenceParams(params:
params)

        navigationController?.pushViewController(conferenceViewController,
animated: true)
    }
}
```

OC

```
#import <TUIRoomKit/TUIRoomKit-Swift.h>

// CreateConferenceViewController 为您自己的 ViewController
@interface CreateConferenceViewController ()

@end

@implementation CreateConferenceViewController

- (void)quickStartConferenceAction {
    // 请替换 "123456" 为您自定义的房间号
    StartConferenceParams *params = [[StartConferenceParams alloc]
initWithRoomId: @"123456"];
    ConferenceMainViewController *conferenceViewController =
[[ConferenceMainViewController alloc] init];
    [conferenceViewController setStartConferenceParams:params];

    [self.navigationController
pushViewController:conferenceViewController animated:YES];
}

@end
```

步骤六：普通成员加入会议

按照如下示例创建并跳转 `ConferenceMainViewController`，即可加入一场会议。

Swift

```
import TUIRoomKit

// EnterConferenceViewController 为您自己的 ViewController
class EnterConferenceViewController: UIViewController {
    private func joinConferenceAction() {
        let params = JoinConferenceParams(roomId: "123456") // 请替换
        "123456" 为您要加入的会议号
        let conferenceViewController = ConferenceMainViewController()
        conferenceViewController.setJoinConferenceParams(params: params)

        navigationController?.pushViewController(conferenceViewController,
            animated: true)
    }
}
```

OC

```
#import <TUIRoomKit/TUIRoomKit-Swift.h>

// EnterConferenceViewController 为您自己的 ViewController
@interface EnterConferenceViewController ()

@end

@implementation EnterConferenceViewController

- (void)joinConferenceAction {
    // 请替换 "123456" 为您自定义的房间号
    JoinConferenceParams *params = [[JoinConferenceParams alloc]
initWithRoomId: @"123456"];
    ConferenceMainViewController *conferenceViewController =
[[ConferenceMainViewController alloc] init];
    [conferenceViewController setJoinConferenceParams:params];

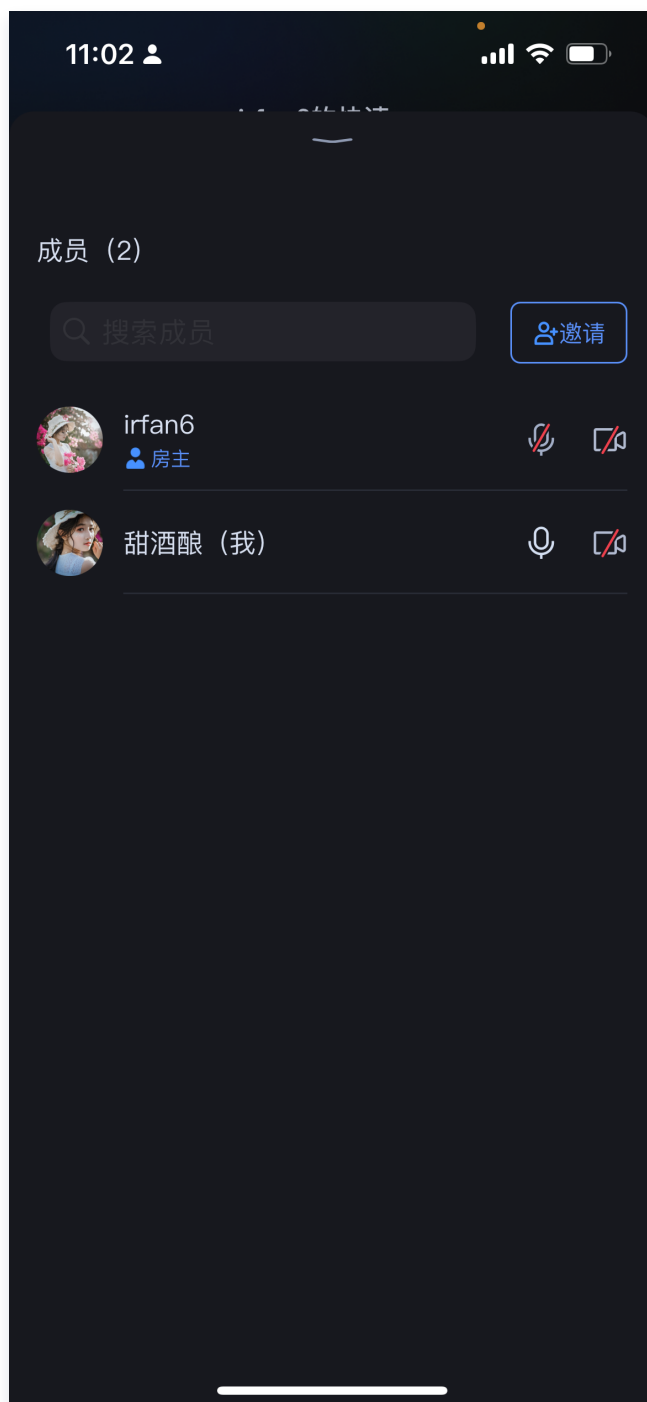
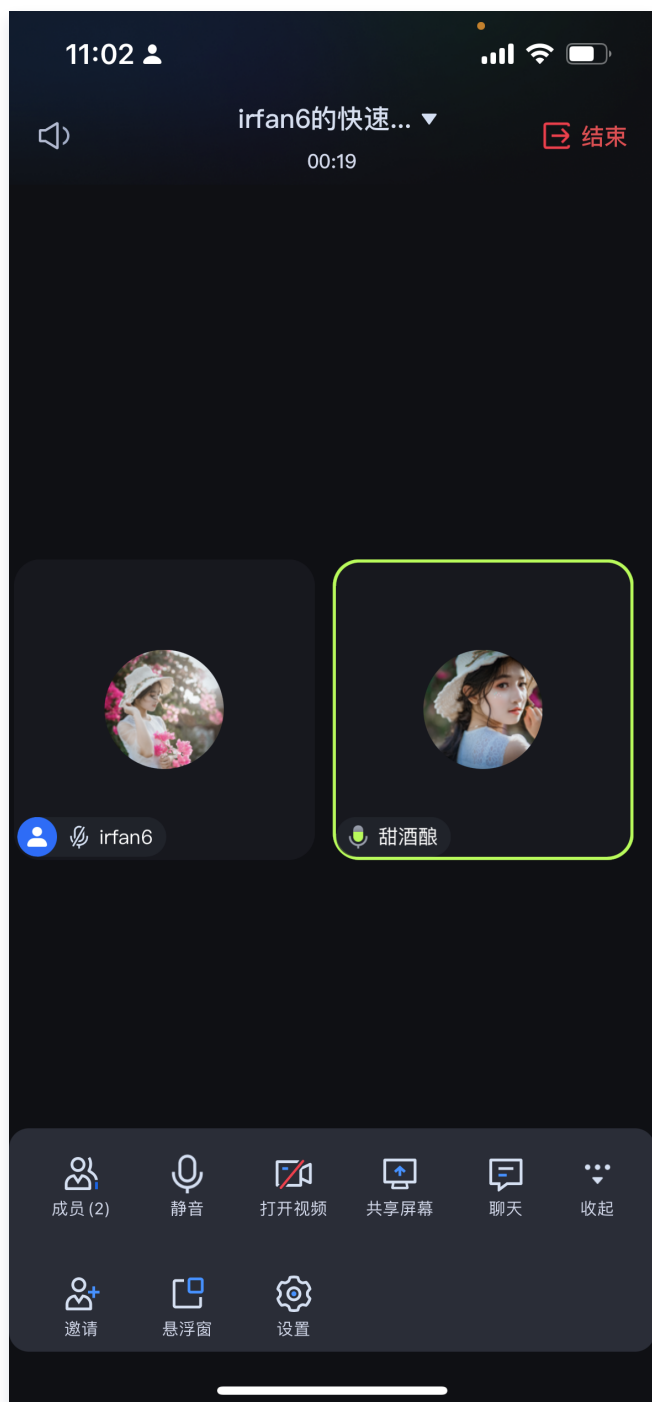
    [self.navigationController
pushViewController:conferenceViewController animated:YES];
}
```

@end

❗ 说明:

当使用OC进行接入时，需要在桥接文件中导入TUIRoomKit。

```
#import <TUIRoomKit/TUIRoomKit-Swift.h>
```



[成员会议主界面](#)[成员用户列表](#)

常见问题

如果您的接入和使用中遇到问题，请参见 [常见问题](#)。

更多特性

- [设置昵称、头像](#)
- [预定会议](#)
- [会中聊天](#)

交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 [zhiliao](#)，进行技术交流和问题反馈。

uni-app（小程序）

最近更新时间：2024-11-28 16:30:42

TUIRoomKit 是腾讯云推出的多人音视频 UI 组件。组件提供房间管理、音视频控制、屏幕共享、成员管理、麦位管理、即时聊天、自定义布局切换等丰富的功能交互。同时支持中英文切换，一键换肤等能力。



本文介绍 TUIRoomKit (Uniapp) 的接入指引，助力您快速上线企业会议、在线教育、医疗问诊、在线巡视、远程定损等业务场景。

您可以单击 [Github](#)/[Gitee](#) 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Uniapp 示例工程。

小程序 Demo

- 如果您想要直接跑通一个新工程，请直接阅读 [uni-app demo 快速跑通](#)。
- 如果您想要亲自集成 TUIRoomKit 组件，搭建一个多人音视频房间小程序，请跟随本文档。

开发环境要求

- 微信 App iOS 最低版本要求：7.0.9。
- 微信 App Android 最低版本要求：7.0.8。
- 小程序基础库最低版本要求：2.13.0。

警告：

- 由于小程序测试号不具备 `<live-pusher>` 和 `<live-player>` 的使用权限，请使用**企业小程序账号**申请相关权限进行开发。
- 由于微信开发者工具不支持原生组件（即 `<live-pusher>` 和 `<live-player>` 标签），需要在**真机**上进行运行体验。

步骤一：开发前准备

1. 开通小程序权限

由于 TUIRoomKit 所使用的小程序标签有更苛刻的权限要求，因此集成 TUIRoomKit 的第一步就是要开通小程序的类目和标签使用权限。**否则无法使用**，这包括如下步骤：

- 小程序推拉流标签不支持个人小程序，只支持企业类小程序。需要在 [注册](#) 时填写主体类型为企业，如下图所示：

注册国家/地区

中国大陆

主体类型

如何选择主体类型？

个人

企业

政府

媒体

其他组织

企业包括：企业、分支机构、个体工商户、企业相关品牌。

主体信息登记

企业类型

☐ 企业 ☐ 个体工商户

企业名称

需与当地政府颁发的商业许可证书或企业注册证上的企业名称完全一致，信息审核成功后，企业名称不可修改。

营业执照注册号

请输入15位营业执照注册号或18位的统一社会信用代码

注册方式

请先填写名称

管理员信息登记

管理员身份证姓名

请填写该小程序管理员的姓名，如果名字包含分隔号“.”，请勿省略。

管理员身份证号码

请输入管理员的身份证号码，一个身份证号码只能注册5个小程序。

管理员手机号码

获取验证码

请输入您的手机号码，一个手机号码只能注册5个小程序。

短信验证码

无法接收验证码？

请输入手机短信收到的6位验证码

管理员身份验证

请先填写企业名称与管理员身份信息

继续

- 小程序推拉流标签使用权限暂时只开放给有限 [类目](#)。
- 符合类目要求的小程序，需要在 [微信公众平台](#) > 开发 > 开发管理 > 接口设置中自助开通该组件权限，如下图所示：

版权所有：腾讯云计算（北京）有限责任公司

第28 共75页



2. 在小程序控制台配置域名

在 [微信公众平台](#) > [开发](#) > [开发管理](#) > [开发设置](#) > [服务器域名](#)中设置 **request 合法域名** 和 **socket 合法域名**，如下图所示：

- **request 合法域名：**

```
https://official.opensso.tencent-cloud.com
https://yun.tim.qq.com
https://cloud.tencent.com
https://webim.tim.qq.com
https://query.tencent-cloud.com
https://web.sdk.qcloud.com
```

- **socket 合法域名：**

❗ 说明：

自 Chat v3.4.6起，SDK 支持独立域名，您需要配置独立域名可更好地保障服务稳定性。

例如您的 SDKAppID 是 1400xxxxxx，则独立域名为：

wss://1400xxxxxxw4c.my-imcloud.com。

```
wss://${SDKAppID}w4c.my-imcloud.com
wss://wss.im.qcloud.com
```

```
wss://wss.tim.qq.com  
wss://wssv6.im.qcloud.com
```

配置服务器信息

① 身份认证 ——— ② 配置服务器信息

如需购买服务器资源及域名，可前往[腾讯云](#)购买。有可使用官方推出的[微信云开发](#)或[微信云托管](#)，无需服务器及域名配置即可上线小程序，[立即开通](#)

request合法域名

https://official.opensso.tencent-cloud.com;https://yun.tim.qq.com;https://cloud.tencent.com;https://webim.tim.qq.com;https://query.tencent-cloud.com;https://web.sdk.qcloud.com;

socket合法域名

wss://wss.im.qcloud.com;wss://wss.tim.qq.com;

uploadFile合法域名

以 https:// 开头。可填写多个域名，域名间请用；分割

downloadFile合法域名

以 https:// 开头。可填写多个域名，域名间请用；分割

udp合法域名

以 udp:// 开头。可填写多个域名，域名间请用；分割

tcp合法域名

以 tcp:// 开头。可填写多个域名，域名间请用；分割

3. 开通服务

在使用 TUIRoomKit 发起会议前，您需要前往控制台开通 TUIRoomKit 专属的多人音视频互动服务，具体步骤请参见 [开通服务](#)。

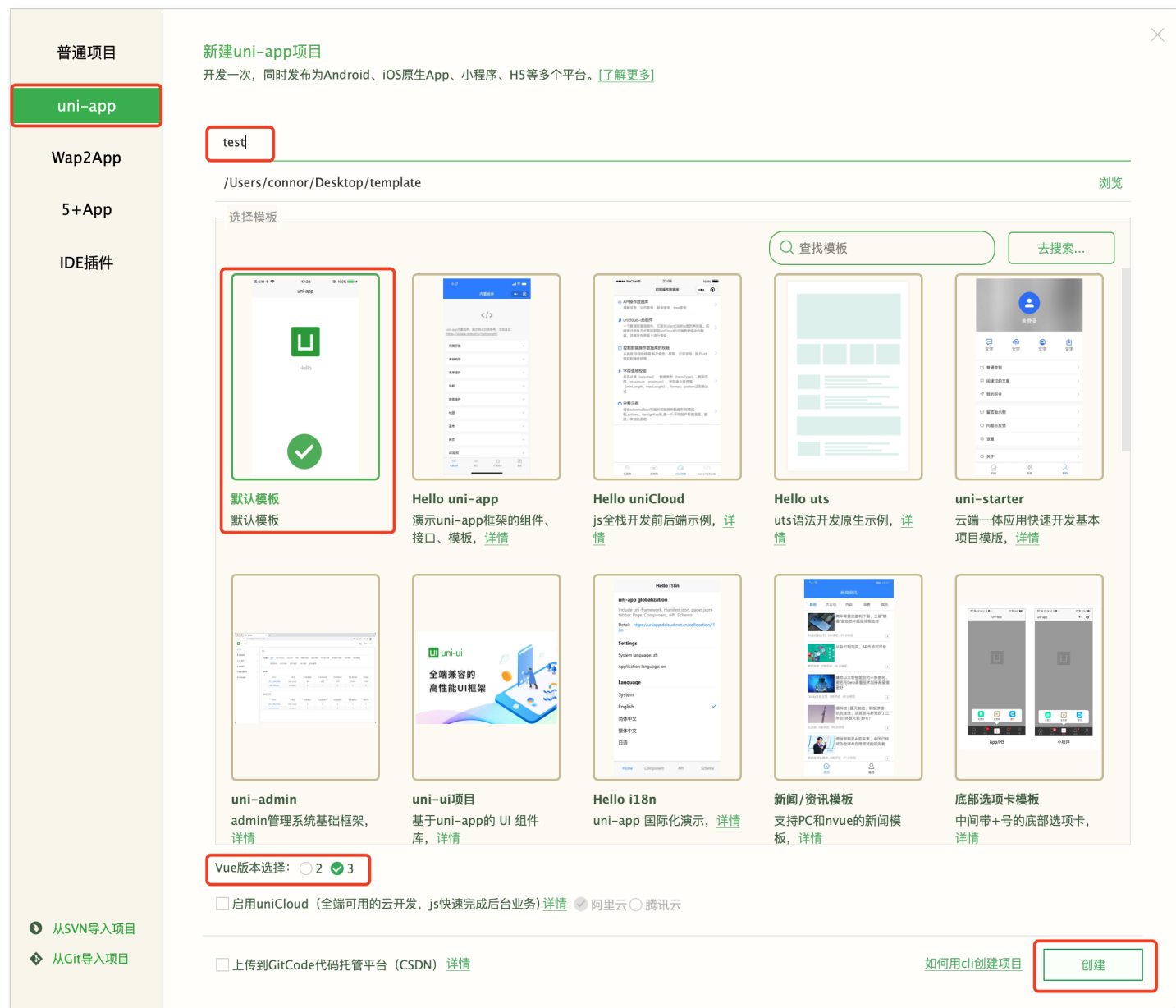
步骤二：准备 Uniapp 工程代码

⚠ 注意：

目前只支持 ts + vite + vue3 工程

如果您已有自己的 uni-app 工程，可跳过该步骤。如果没有，可选择以下方式生成模板工程。

在 HBuilder 中创建小程序项目。



步骤三：下载并导入 TUIRoom 组件

⚠ 注意：

因为小程序对主包大小的限制，这里 TUIRoom 默认使用分包方案，通过脚本自动配置。

3.1 npm 依赖安装

此处需在 uni-app 项目根目录下运行

```
# 在终端输入npm init -y, 创建package.json文件。
npm init -y
# 安装项目依赖
npm i @tencentcloud/tui-roomkit-uniapp pinia
```

3.2 执行项目自动化脚本

```
# 此脚本会自动拷贝相关依赖到您的项目。并配置 vite.config.ts、tsconfig.json、
manifest.json、pages.json 文件
# 同时创建一个空白页面 roomkitTest.vue 用于展示 roomKit 项目
node ./node_modules/@tencentcloud/tui-roomkit-uniapp/configureProject.js
```

警告：

此处如果没有特殊需求建议您选择 **自动配置**，如执行脚本的过程中您未选择自动配置，请结合 [项目配置说明](#) 手动进行修改。

3.3 手动修改项目配置

1. 修改 `./src/main.js (main.ts)`，引入 `pinia`。（注意此处为增量添加，根据自己项目选择添加位置）

注意：

如果项目使用 HbuilderX 创建，main.js 文件会有条件编译相关代码。需要您检查是否将配置正确填写在了 `#ifdef VUE3` 条件中。

HbuilderX

```
import App from './App'
import { createPinia } from "pinia";

// #ifndef VUE3
import Vue from 'vue'
import './uni.promisify.adaptor'
Vue.config.productionTip = false
App.mpType = 'app'
const app = new Vue({
  ...App
})
app.$mount()
app.use(createPinia());
```

```
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue'
export function createApp() {
  const app = createSSRApp(App)
  app.use(createPinia());
  return {
    app
  }
}
// #endif
```

CLI

```
import { createPinia } from 'pinia';
// locales 为 TUIRoom/locales，因为主包无法使用分包资源，此处拷贝了一份 locales
在 src 路径下

app.use(createPinia());
```

2. 找到并打开 `./roomkit/config/basic-info-config.js` 文件，并修改 `basic-info-config.js` 文件中的相关参数：

```
/**
 * Tencent Cloud SDKAppId, which should be replaced with user's SDKAppId.
 * Enter Tencent Cloud TRTC [Console] (https://console.cloud.tencent.com/trtc) to create an application,
 * and you will see the SDKAppId.
 * It is a unique identifier used by Tencent Cloud to identify users.
 *
 * 腾讯云 SDKAppId, 需要替换为您自己账号下的 SDKAppId。
 * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav) 创建应用, 即可看到 SDKAppId,
 * 它是腾讯云用于区分客户的唯一标识。
 */

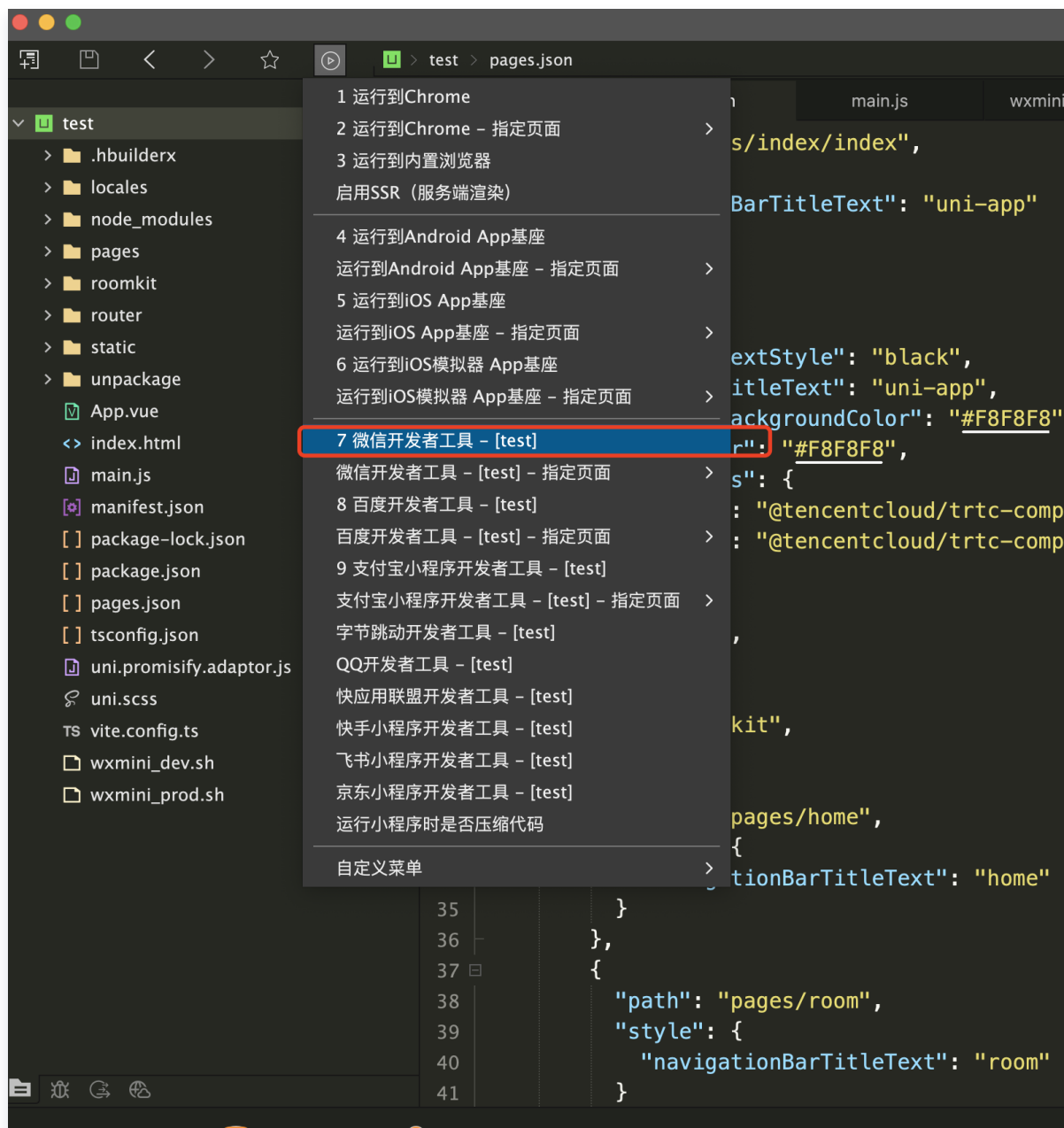
export const SDKAPPID = 0;

/**
 * Encryption key for calculating signature, which can be obtained in the following steps:
 *
 * Step1. Enter Tencent Cloud TRTC [Console](https://console.cloud.tencent.com/rav),
 * and create an application if you don't have one.
 * Step2. Click your application to find "Quick Start".
 * Step3. Click "View Secret Key" to see the encryption key for calculating UserSig,
 * and copy it to the following variable.
 *
 * Notes: this method is only applicable for debugging Demo. Before official launch,
 * please migrate the UserSig calculation code and key to your backend server to avoid
 * unauthorized traffic use caused by the leakage of encryption key.
 * Document: https://intl.cloud.tencent.com/document/product/647/35166#Server
 *
 * 计算签名用的加密密钥, 获取步骤如下:
 *
 * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav), 如果还没有应用就创建一个,
 * step2. 单击“应用配置”进入基础配置页面, 并进一步找到“帐号体系集成”部分。
 * step3. 点击“查看密钥”按钮, 就可以看到计算 UserSig 使用的加密的密钥了, 请将其拷贝并复制到如下的变量中
 *
 * 注意: 该方案仅适用于调试Demo, 正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上, 以避免加密密钥泄露导致的流量盗用。
 * 文档: https://cloud.tencent.com/document/product/647/17275#Server
 */

export const SDKSECRETKEY = '';
```

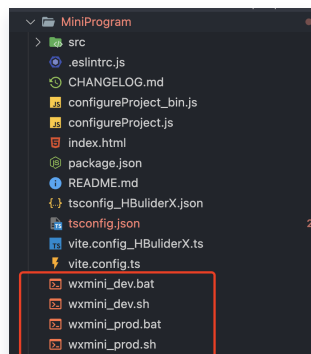
3.4 运行到微信开发者工具

1. 运行到微信开发者工具, 等待编译完成并打开微信开发者工具。



2. 打开微信开发者工具后，在项目根目录（非编译后 dist 文件目录）执行脚本，install 相关依赖。

若项目中脚本不存在，可手动下载脚本并将其拷贝到您的项目根目录：[wxmini_uniapp_script](#)。



Windows

```
# npm run dev:mp-weixin 编译完成后执行该指令
# 此处使用的路径为默认路径，如果您对此路径进行了修改，需要手动对脚本路径进行调整
./wxmini_dev.bat
```

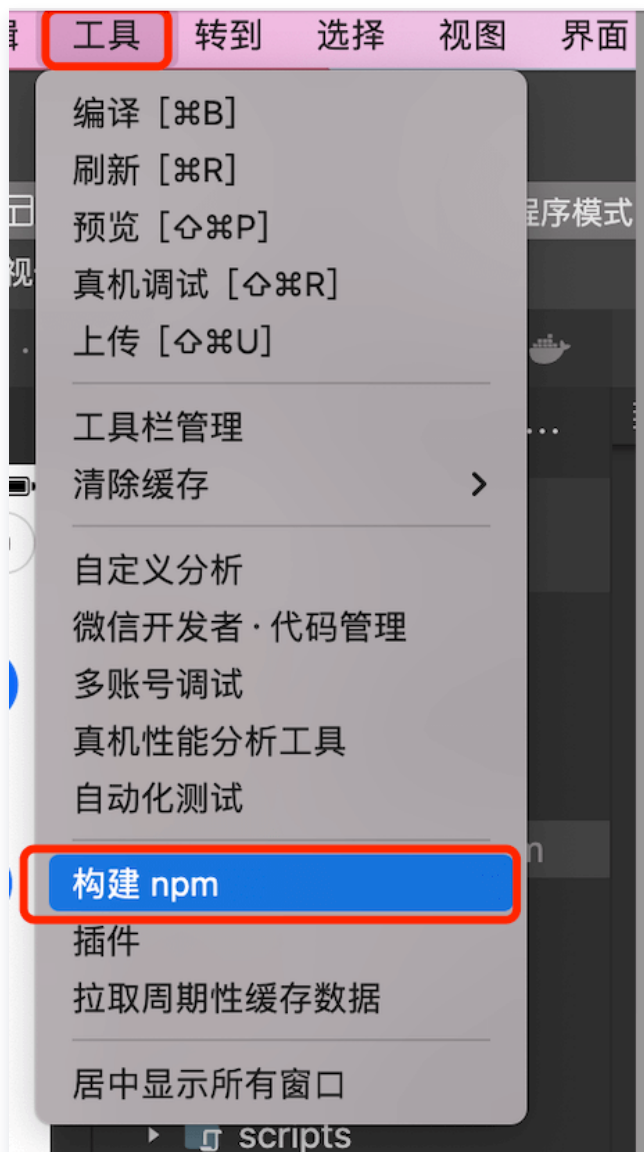
Mac

```
# npm run dev:mp-weixin 编译完成后执行该指令
# 此处使用的路径为默认路径，如果您对此路径进行了修改，需要手动对脚本路径进行调整
bash wxmini_dev.sh
```

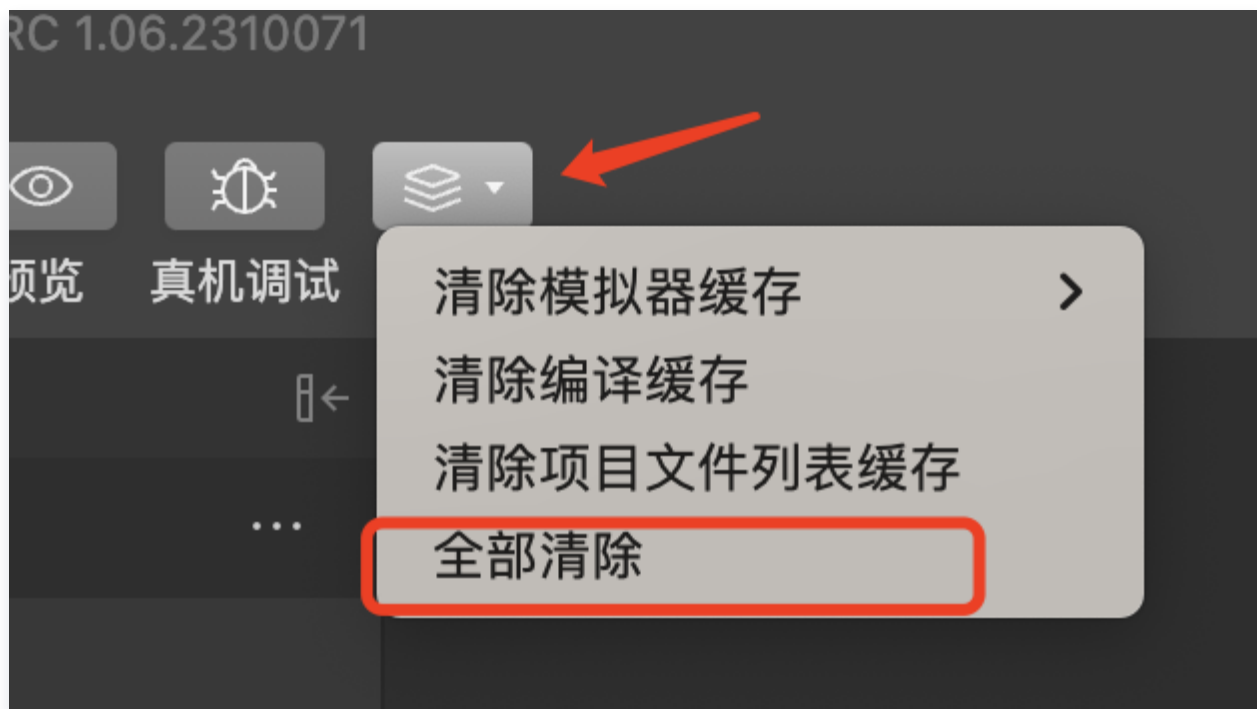
❗ 说明:

因为小程序对主包大小有限制，这里默认采用了分包方案，部分依赖需要进入编译后的目录执行 `npm install` 进行下载，此处将操作指令集成在 `wxmini_dev.sh/wxmini_dev.bat` 脚本中，因此需要在编译完成后执行 `bash wxmini_dev.sh/wxmini_dev.bat` 安装依赖，您也可以根据指令内容手动安装。

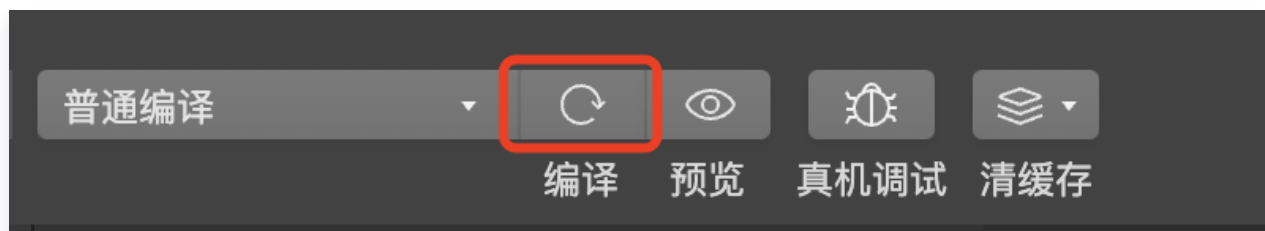
3. 构建 npm，微信开发者工具单击工具 > 构建 npm。具体如下图：



4. 单击清除缓存 > 全部清除，避免开发者工具的缓存造成渲染异常。



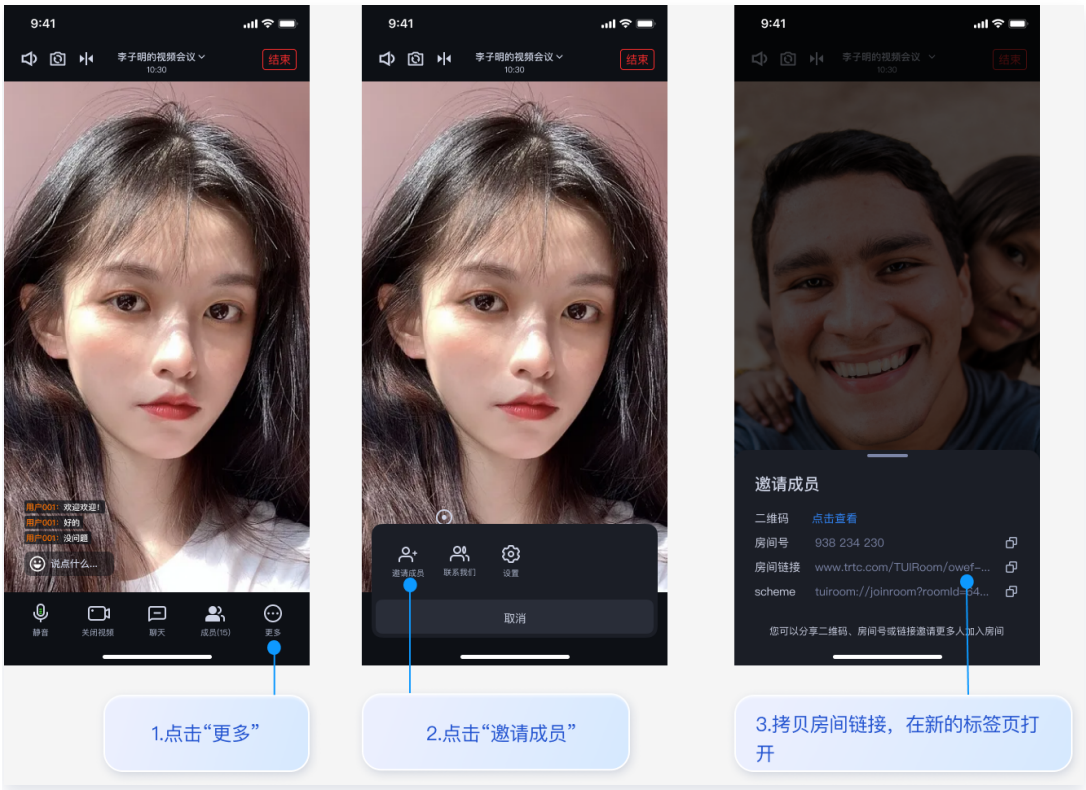
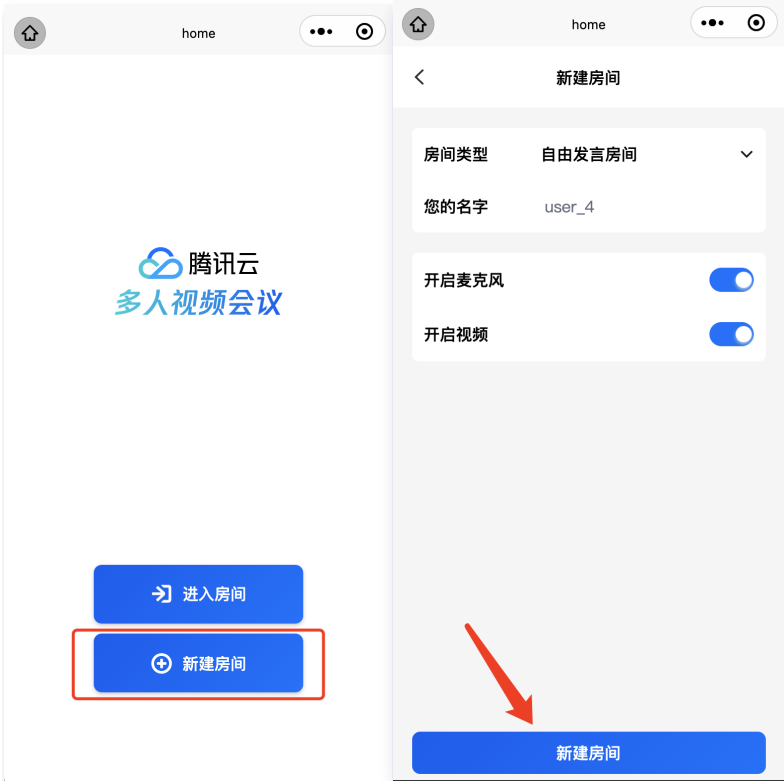
5. 编译小程序。



6. 点击预览，扫描二维码，体验程序。

步骤四：开始您的第一场会议

单击新建房间，开始您的第一场会议。



项目配置说明

警告：

以下修改均为增量修改，需要根据自己项目配置进行添加。

1. 修改 `./src/manifest.json` 小程序相关配置，开启分包并设置 `node_modules` 目录，参考文档 [自定义 node-modules、miniprogram-npm 位置构建](#)、[uniapp 分包](#)。

```
/* 小程序特有相关 */
"mp-weixin": {
  "setting": {
    "urlCheck": false,
    "packNpmManually": true,
    "packNpmRelationList": [
      {
        "packageJsonPath": "./package.json",
        "miniprogramNpmDistDir": "./"
      },
      {
        "packageJsonPath": "./roomkit/package.json",
        "miniprogramNpmDistDir": "./roomkit"
      }
    ]
  },
  "usingComponents": true,
  "optimization": {
    "subPackages": true
  }
},
```

2. 修改 `./src/pages.json`，添加分包页面以及全局组件引入。

```
{
  "pages": [
    {
      "path": "pages/roomkitTest",
      "style": {
        "navigationBarTitleText": ""
      }
    },
  ],
  "subpackages": [
    {
      "root": "roomkit",
      "pages": [
        {
          "path": "pages/home",
```

```
      "style": {
        "navigationBarTitleText": "home"
      }
    },
    {
      "path": "pages/room",
      "style": {
        "navigationBarTitleText": "room"
      }
    }
  ]
},
"globalStyle": {
  "usingComponents": {
    "trtc-pusher": "@tencentcloud/trtc-component-wx/trtc-pusher",
    "trtc-player": "@tencentcloud/trtc-component-wx/trtc-player"
  }
},
}
```

3. 修改 `./vite.config.ts`，将 `@tencentcloud/tuiroom-engine-wx` 设置为外部依赖，用于分包减少主包体积。

HBuliderX 创建

```
import { defineConfig } from 'vite';
import uni from '@dcloudio/vite-plugin-uni';

import path from 'path';

function resolve(dir) {
  return path.join(__dirname, dir);
}

// https://vitejs.dev/config/
export default defineConfig(({ mode }) => {
  const isProd = mode === 'production';
```

```
const commonConfig = {
  optimizeDeps: {
    include: ['@tencentcloud/tuiroom-engine-wx'],
  },
  plugins: [
    uni(),
  ],
  resolve: {
    alias: {
      '@': resolve('./'),
      '@TUIRoom': resolve('/roomkit/TUIRoom'),
    },
  },
  build: {
    rollupOptions: {
      external: ['@tencentcloud/tuiroom-engine-wx'],
    },
  },
};

const devConfig = {
  ...commonConfig,
};

const prodConfig = {
  ...commonConfig,
};

return isProd ? prodConfig : devConfig;
});
```

CLI 创建

```
import { defineConfig } from 'vite';
import uni from '@dcloudio/vite-plugin-uni';

import path from 'path';
```

```
function resolve(dir) {
  return path.join(__dirname, dir);
}

// https://vitejs.dev/config/
export default defineConfig(({ mode }) => {
  const isProd = mode === 'production';

  const commonConfig = {
    optimizeDeps: {
      include: ['@tencentcloud/tuiroom-engine-wx'],
    },
    plugins: [
      uni(),
    ],
    resolve: {
      alias: {
        '@': resolve('src'),
        '@TUIRoom': resolve('src/roomkit/TUIRoom'),
      },
    },
    build: {
      rollupOptions: {
        external: ['@tencentcloud/tuiroom-engine-wx'],
      },
    },
  };

  const devConfig = {
    ...commonConfig,
  };

  const prodConfig = {
    ...commonConfig,
  };

  return isProd ? prodConfig : devConfig;
});
```

4. 修改 `./tsconfig.json`，项目使用 `ts + vue3` 开发，如不存在该文件，会产生编译错误。

HBuliderX 创建

```
// tsconfig.json
{
  "compilerOptions": {
    // 与 Vue 的浏览器支持保持一致
    "target": "es5",
    // 这可以对 `this` 上的数据 property 进行更严格的推断
    "strict": true,
    // 如果使用 webpack 2+ 或 rollup, 可以利用 tree-shake:
    "module": "es2015",
    "moduleResolution": "node"
  }
}
```

CLI 创建

```
// tsconfig.json
{
  "extends": "@vue/tsconfig/tsconfig.json",
  "compilerOptions": {
    "sourceMap": true,
    "baseUrl": ".",
    "paths": {
      "@/*": ["/src/*"]
    },
    "lib": ["esnext", "dom"],
    "types": ["@dcloudio/types"]
  },
  "include": ["src/**/*.ts", "src/**/*.d.ts", "src/**/*.tsx",
    "src/**/*.vue"]
}
```

常见问题

[常见问题](#)

其他文档

- [TUIRoomKit](#)

- [TUIRoom demo 快速跑通](#)
- [设置昵称、头像](#)

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 [腾讯云实时音视频 TRTC 技术交流群](#) 进行技术交流和问题反馈。

uni-app (客户端)

最近更新时间：2025-04-22 10:20:42

TUIRoomKit 是腾讯云推出的多人音视频 UI 组件。组件提供房间管理、音视频控制、屏幕共享、成员管理、麦位管理、即时聊天、自定义布局切换等丰富的功能交互。同时支持中英文切换，一键换肤等能力。



本文介绍 TUIRoomKit (Uniapp) 的接入指引，助力您快速上线企业会议、在线教育、医疗问诊、在线巡视、远程定损等业务场景。

您可以单击 [Github](#) / [Gitee](#) 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Uniapp (客户端) 示例工程。

TUIRoomKit Demo 体验

如果您想要直接跑通一个新工程，请直接阅读 [uni-app demo 快速跑通](#)。

开发环境要求

- HbuilderX 版本要求：HbuilderX 版本 $\geq$ 3.94。
- 插件调试说明：原生插件暂不支持模拟器调试。
- iOS 设备要求：iOS 系统 $\geq$ 9.0，支持音视频通话的真机设备。

- Android 设备要求：Android 系统 $\geq$ 5.0（SDK API Level 21），支持音视频的真机设备，[允许 USB 调试](#)。

步骤一：开通服务

在使用腾讯云提供的音视频服务前，您需要前往控制台，为应用开通音视频服务。具体步骤请参见 [开通服务](#)。

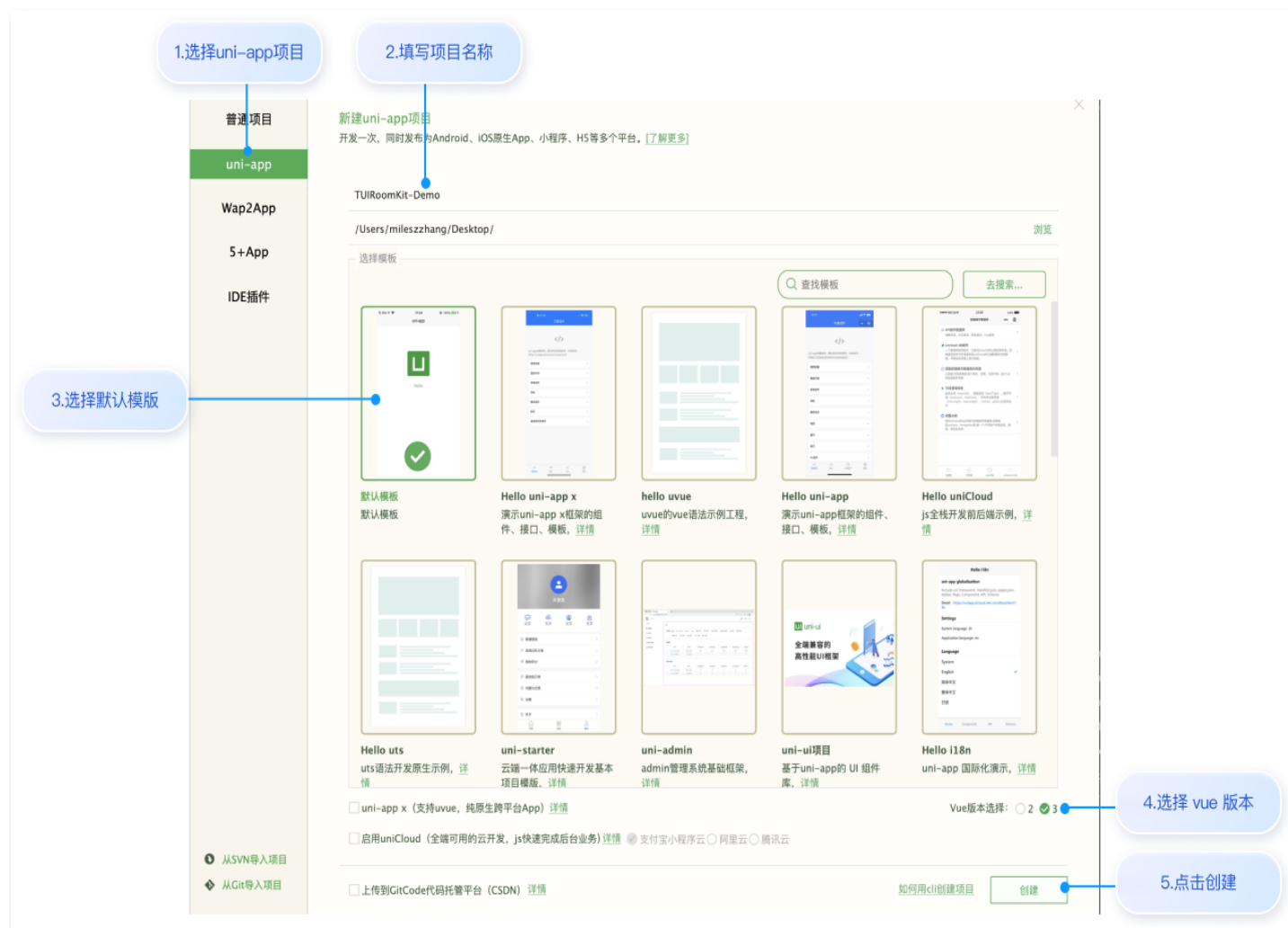
步骤二：创建 uni-app 项目

打开 Hbuilderx 开发工具，点击新建 uni-app 项目：项目名称（TUIRoomKit-Demo）。

⚠ 注意：

目前只支持 ts + vite + vue3 工程

如果您已有自己的 uni-app 工程，可跳过该步骤。如果没有，可选择以下方式生成模板工程。



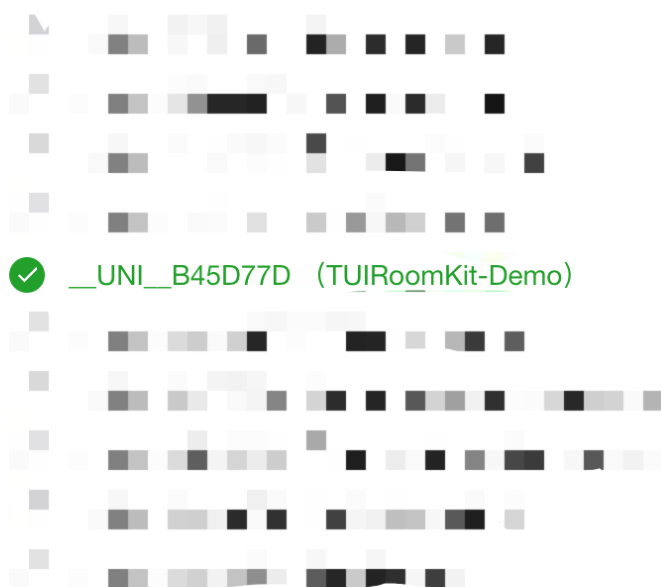
步骤三：下载并导入 TUIRoomEngine 插件

1. 访问 [TencentCloud-TUIRoomEngine 插件](#)，在插件详情页中购买插件，购买插件时选择对应的 AppID，绑定正确的包名。

您要为哪个项目购买本插件？

注意事项：

- 1、一次购买仅用于一个项目。仅与项目有关，与项目有几位开发者无关。
- 2、购买后，在HBuilderX中打开该项目的manifest文件，并在“App原生插件配置”中配置，然后云打包生效。
- 3、暂不支持离线打包集成付费原生插件。



[管理我的APP](#) ☒ 同意《付费插件使用协议》(更新于2023.4.28)

取消

下一步

2. 在 `TUIRoomKit-Demo` 项目中导入插件。



步骤四：下载并导入 TUIRoomKit 组件

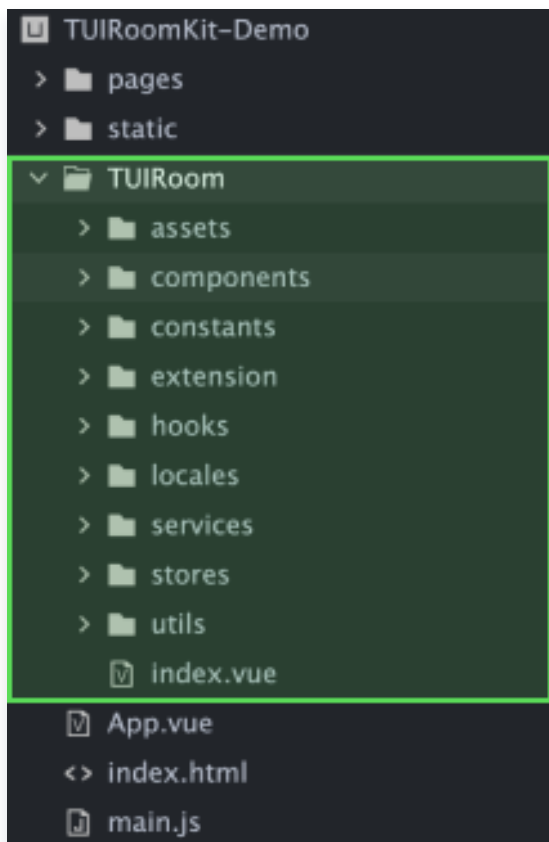
1. 下载 github 源码, 建议下载到和步骤一创建的项目同级目录

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

2. 手动拷贝 TUIRoom 组件至自己的项目中

拷贝 TUIRoomKit/uni-app/src/roomkit/ 下的 TUIRoom 文件夹至创建的项目中

拷贝完目录结构如下：



3. 安装依赖

HBuilderX 创建的工程不会默认创建 package.json 文件，因此您需要先创建 package.json 文件。新建终端，如下：

```
# 在终端输入npm init -y，创建package.json文件。
npm init -y
# 安装项目依赖
npm i @tencentcloud/tuiroom-engine-uniapp-app @tencentcloud/universal-api mitt pinia --save
```

4. 项目工程配置

注册 pinia: TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia。项目入口文件为 TUIRoomKit-Demo/main.js 文件。

```
// #ifndef VUE3
import Vue from 'vue'
import './uni.promisify.adaptor'
Vue.config.productionTip = false
App.mpType = 'app'
const app = new Vue({
  ...App
})
```

```
app.$mount()
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue';
import App from '@App.vue';
import * as Pinia from 'pinia';
export function createApp() {
  const app = createSSRApp(App);
  app.use(Pinia.createPinia());
  return {
    app,
    Pinia,
  };
}
// #endif
```

配置 pages.json: 为了更好的使用 TUIRoom 组件的能力，建议您在 `TUIRoomKit-Demo/pages.json` 中新增如下配置。

```
"navigationStyle": "custom",
"app-plus": {
  "titleNView": false
}
```

```
{
  "pages": [ //pages数组中第一项表示应用启动页，参考：https://uniapp.dcloud.io/collocation/pages
    {
      "path": "pages/index/index",
      "style": {
        "navigationBarTitleText": "uni-app",
        "navigationStyle": "custom",
        "app-plus": {
          "titleView": false
        }
      }
    }
  ],
  "globalStyle": {
    "navigationBarTextStyle": "black",
    "navigationBarTitleText": "uni-app",
    "navigationBarBackgroundColor": "#F8F8F8",
    "backgroundColor": "#F8F8F8"
  },
  "uniIdRouter": {}
}
```

5. 模块配置

安卓需要使用到麦克风权限，所以需要在 App 模块配置中添加录音模块。



6. 引用 TUIRoom 组件, 填写 sdkAppId 等信息。

⚠ 注意:

由于原生插件只可以在 `nvue` 页面中使用, 所以您需要将引入 TUIRoom 组件的页面更改为 `nvue` 页面, 详情见 [nvue 使用场景](#)。

- TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 `init`、`createRoom`、`enterRoom` 方法。
- 主持人及普通成员可通过 `init` 方法向 TUIRoom 组件初始化应用及用户数据, 主持人可通过 `createRoom` 方法创建并加入房间, 普通成员可通过 `enterRoom` 方法加入主持人已经创建好的房间。

在 `pages/index/index.nvue` 页面中引入 TUIRoom 组件

⚠ 注意:

请务必将 `pages/index/index.vue` 修改为 `pages/index/index.nvue`。

```
<template>
  <room ref="TUIRoomRef" ></room>
</template>
```

```
<script setup lang="ts">
import { ref, onMounted } from 'vue';
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from '../../TUIRoom/index.vue';
// 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
const TUIRoomRef = ref();

onMounted(async () => {
  // 初始化 TUIRoom 组件
  // 主持人在创建房间前需要先初始化 TUIRoom 组件
  // 普通成员在进入房间前需要先初始化 TUIRoom 组件
  await TUIRoomRef.value.init({
    // 获取 sdkAppId 请您参考 步骤一
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
    userId: '',
    // 本地开发调试可在
https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig,
    注意 userSig 与 userId 为一一对应关系
    userSig: '',
    // 用户在您业务中使用的昵称
    userName: '',
    // 用户在您业务中使用的头像链接
    avatarUrl: '',
  })
  // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
  await handleCreateRoom();
})

// 主持人创建房间，该方法只在创建房间时调用
async function handleCreateRoom() {
  // roomId 为用户进入的房间号，要求 roomId 类型为 string
  // roomMode 包含 'FreeToSpeak' (自由发言模式) 和 'SpeakAfterTakingSeat' (上台发言模式) 两种模式，默认为 'FreeToSpeak'，注意目前仅支持自由发言模式
  // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id）
  const roomId = '123456';
  const roomMode = 'FreeToSpeak';
  const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
  }
}
```

```
try {
    await TUIRoomRef.value.createRoom({ roomId, roomName: roomId,
roomMode, roomParam });
} catch (error: any) {
    alert('TUIRoomKit.createRoom error: ' + error.message);
}
}

// 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
async function handleEnterRoom() {
    // roomId 为用户进入的房间号，要求 roomId 类型为 string
    // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id）
    const roomId = '123456';
    const roomParam = {
        isOpenCamera: true,
        isOpenMicrophone: true,
    }
    try {
        await TUIRoomRef.value.enterRoom({ roomId, roomParam });
    } catch (error: any) {
        alert('TUIRoomKit.enterRoom error: ' + error.message);
    }
}
</script>
```

步骤五：开启您的第一场会议

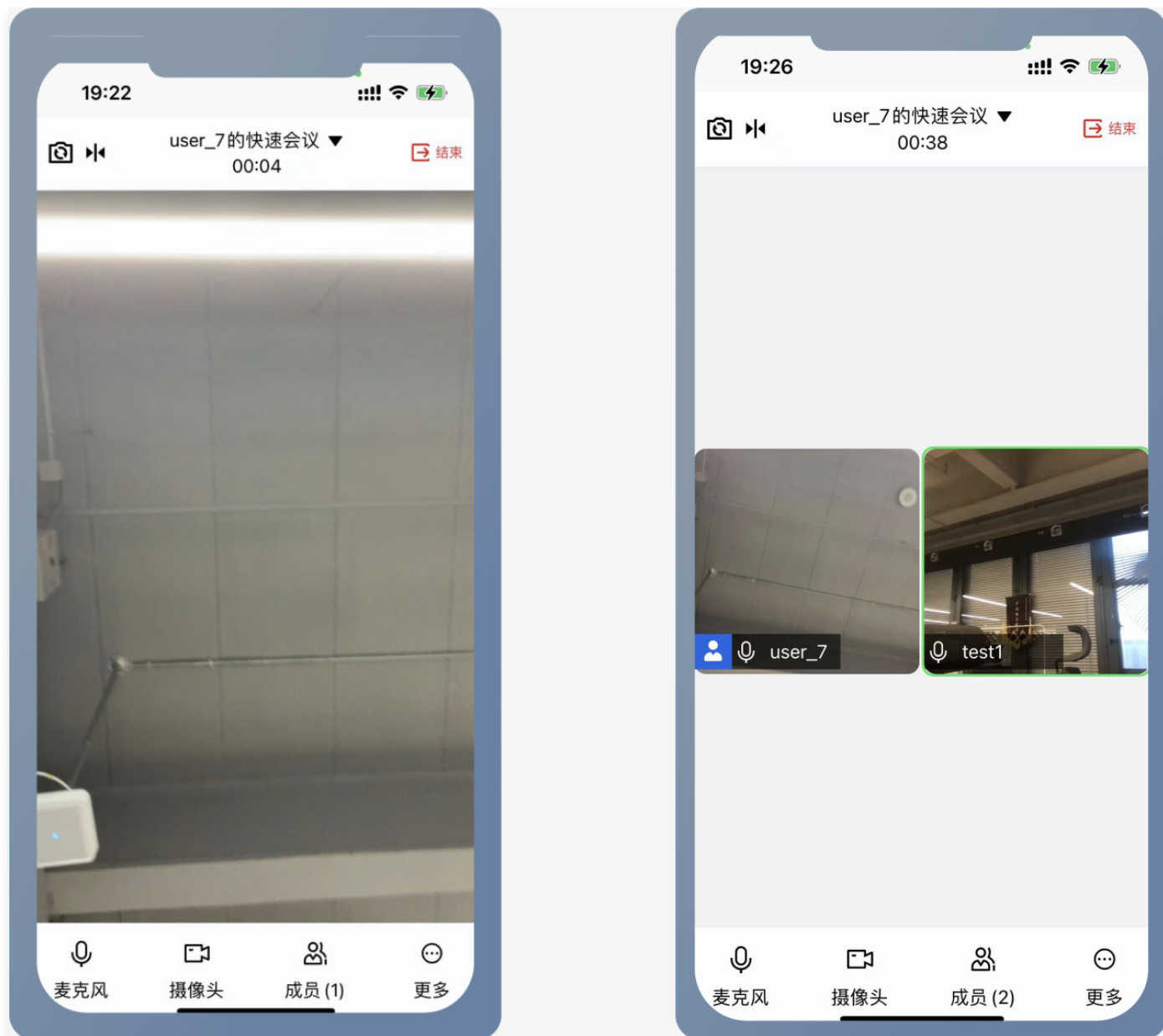
1. 制作自定义调试基座，请选择传统打包方式进行打包。



2. 自定义调试基座成功后，使用自定义基座运行项目。



3. 主持人端创建会议以及普通成员加入会议界面。



常见问题

1. 若安卓平台，跑通出现 “reportJSException >>>> exception function:createInstanceContext, exception:white screen cause create instanceContext failed,check js stack ->Uncaught TypeError: (0, import pinia.defineStore) is not a function” 错误，如何解决？

```
18:07:03.542 reportJSException >>>> exception function:createInstanceContext, exception:white screen cause create instanceContext failed,check js stack ->Uncaught TypeError: (0, import pinia.defineStore) is not a function
at (pages/index/index.js:1366:52)
at (pages/index/index.js:10753:3)
```

可能是由于 Pinia 版本与 HBuilder 版本不兼容导致的，详情参见 [状态管理 Pinia](#)。

2. 若 iOS 平台打包自定义基座时，出现体积超过40M的提示。如何解决？

这是和其他原生插件一块使用时造成的体积超出，详情见：[本地 uni-app 原生插件提交云端打包](#)。

更多特性

- [设置昵称、头像](#)

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 [腾讯云实时音视频 TRTC 技术交流群](#) 进行技术交流和问题反馈。

Flutter

最近更新时间：2024-09-14 10:56:11

本文将指导您如何在快速完成 `TUIRoomKit` 组件的接入。遵循本文档的步骤，您将在一小时内完成以下关键步骤，并最终实现一个具备完整 UI 界面的音视频会议功能。

环境准备

平台	版本
Flutter	3.7.0 及以上版本。
Android	- 最低兼容 Android 4.1 (SDK API Level 16) ，建议使用 Android 5.0 (SDK API Level 21) 及以上版本。 - Android Studio 3.5 及以上的版本 (Gradle 3.5.4 及以上的版本) 。 - Android 4.1 及以上的手机设备。
iOS	iOS 12.0 及更高。

步骤一：开通服务

在使用 `TUIRoomKit` 发起会议前，您需要前往控制台开通 `TUIRoomKit` 专属的多人音视频互动服务，具体步骤请参见 [开通服务](#)。

步骤二：集成 `TUIRoomKit` 组件

在您的工程 `pubspec.yaml` 文件中，添加 `tencent_conference_uikit` 插件依赖。

```
dependencies:  
  tencent_conference_uikit: 最新版本
```

执行以下命令安装组件：

```
flutter pub get
```

步骤三：完成工程配置

- 由于 `tencent_conference_uikit` 组件使用了 `GetX` 状态管理库的相关功能，您需要在您的应用程序中使用 `GetMaterialApp` 来代替 `MaterialApp`。或者您也可以将您 `MaterialApp` 中的 `navigatorKey` 属性设置为 `Get.key` 以实现相同的效果。

//此步骤需导入get包后再操作。由于tencent_conference_uikit中已有对get的依赖，您无需再您的pubspec.yaml中进行额外的配置。

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GetMaterialApp( // 使用GetMaterialApp来代替MaterialApp
      // 您原先的MaterialApp内容
    );
  }
}
```

- 使用 **Xcode** 打开您的工程，选择项目 > **Building 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。
- 如果您需要在 **iOS** 端使用音视频功能，需要授权麦克风和摄像头的使用权限（**Android** 端已在 **SDK** 中声明相关权限，您无需手动进行相关配置）。

在 App 的 **Info.plist** 中添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

```
<key>NSCameraUsageDescription</key>
<string>TUIRoom需要访问您的相机权限</string>
<key>NSMicrophoneUsageDescription</key>
<string>TUIRoom需要访问您的麦克风权限</string>
```

完成以上添加后，在您的 **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
```

步骤四：登录组件

在您的项目中添加如下代码，它的作用是通过调用 `TUIRoomKit` 中的相关接口完成组件的初始化。这个步骤非常关键，因为只有在初始化后才能正常使用 `TUIRoomKit` 的各项功能，故请您耐心检查相关参数是否配置正确：

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

var result = await TUIRoomEngine.login(
  SDKAPPID, // 请替换为您的SDKAPPID
  'userId', // 请替换为您的UserID
  'userSig', // 请替换为您的userSig
);

if (result.code == TUIError.success) {
  // login success
} else {
  // login error
}
```

参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- **SDKAppID**：在 [步骤一](#) 中的第3步中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**：使用 [步骤一](#) 的第3步中获取的 SDKSecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 [辅助工具](#) 生成一个临时可用的 UserSig。

更多信息请参见 [如何计算及使用 UserSig](#)。

⚠ 注意：

- 这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：
 - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
 - UserSig 被错配成了加密密钥（SDKSecretKey），UserSig 是用 SDKSecretKey 把 SDKAppID、UserID 以及过期时间等信息加密得来的，而不是直接把 SDKSecretKey 配置成 UserSig。
 - UserID 被设置成“1”、“123”、“111”等简单字符串，由于 TRTC 不支持同一个 UserID 多端登录，所以在多人协作开发时，形如“1”、“123”、“111”这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。
- 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 SDKSecretKey 暴露在 App 的代码当中，这并不利于您后续升

级和保护您的 SDKSecretKey，所以我们强烈建议您将 UserSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUIRoomKit 组件时向您的服务器请求实时计算出的 UserSig。

登录弹幕聊天（可选）

Flutter TUIRoomKit（tencent_conference_uikit）自2.4.1版本开始引入弹幕聊天功能。如您需要使用弹幕聊天功能，需要完成以下初始化及登录。（如您还需要使用 [会中聊天](#) 页面，可忽略此步骤，完成会中聊天的 [初始化及登录](#) 即可）：

```
import 'package:tencent_cloud_chat_sdk/enum/V2TimSDKListener.dart';
import 'package:tencent_cloud_chat_sdk/enum/log_level_enum.dart';
import 'package:tencent_cloud_chat_sdk/models/v2_tim_callback.dart';
import 'package:tencent_cloud_chat_sdk/tencent_im_sdk_plugin.dart';

// 初始化
var initResult = await TencentImSDKPlugin.v2TIMManager.initSDK(
  sdkAppID: SDKAPPID, // 请替换为您的SDKAPPID
  loglevel: LogLevelEnum.V2TIM_LOG_INFO, // 日志登记等级
  listener: V2TimSDKListener(), // 事件监听器。使用弹幕聊天时，这里传空对象即可。
);

if (initResult.code == 0) { // 初始化成功
  // 登录
  V2TimCallback imLoginResult = await
  TencentImSDKPlugin.v2TIMManager.login(
    userID: 'userId', // 请替换为您的UserID
    userSig: 'userSig', // 请替换为您的userSig
  );
}
```

❗ 说明：

弹幕聊天功能在 TUIRoomKit 中默认开启。如您无需弹幕聊天功能，您无需进行上述初始化及登录操作，并可以通过底部栏 > 设置 > 开启弹幕聊天选项来关闭弹幕聊天。

步骤五：使用组件

设置用户信息（可选）

通过调用 TUIRoomEngine 的 setSelfInfo 即可设置当前用户的用户名、头像。

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

TUIRoomEngine.setSelfInfo(userName, avatarURL);
```

参数	类型	含义
userName	String	用户名
avatarURL	String	用户头像 URL

开始快速会议

通过调用 `ConferenceSession` 的 `quickStart` 方法，便可以开始快速会议。

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

ConferenceSession.newInstance('roomId') //您的room id
  ..onActionSuccess = _quickStartSuccess
  ..onActionError = _quickStartError
  ..quickStart();

void _quickStartSuccess() {
  //您可以在开始快速会议的成功回调中, 自行导航至会议页面。
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => ConferenceMainPage(),
    ),
  );
}

void _quickStartError(ConferenceError error, String message) {
  debugPrint("code: $error message: $message");
}
```

加入会议

通过调用 `ConferenceSession` 的 `join` 方法，便可以加入指定会议。

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

ConferenceSession.newInstance('roomId') //您的room id
```

```
..onActionSuccess = _joinSuccess
..onActionError = _joinError
..join();

void _joinSuccess() {
  //您可以在加入会议的成功回调中,自行导航至会议页面。
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => ConferenceMainPage(),
    ),
  );
}

void _joinError(ConferenceError error, String message) {
  debugPrint("code: $error message: $message");
}
```

❗ 说明:

上述的示例代码中,只展示了最简单的创建会议/加入会议的方式。如果您需要在进会前完成更多设置,请参见 [会前控制](#)。

运行

运行代码后,您将看到如下的会议界面。



更多功能

- [设置昵称、头像](#)
- [预定会议](#)
- [会中聊天](#)
- [RoomKit API](#)

交流与反馈

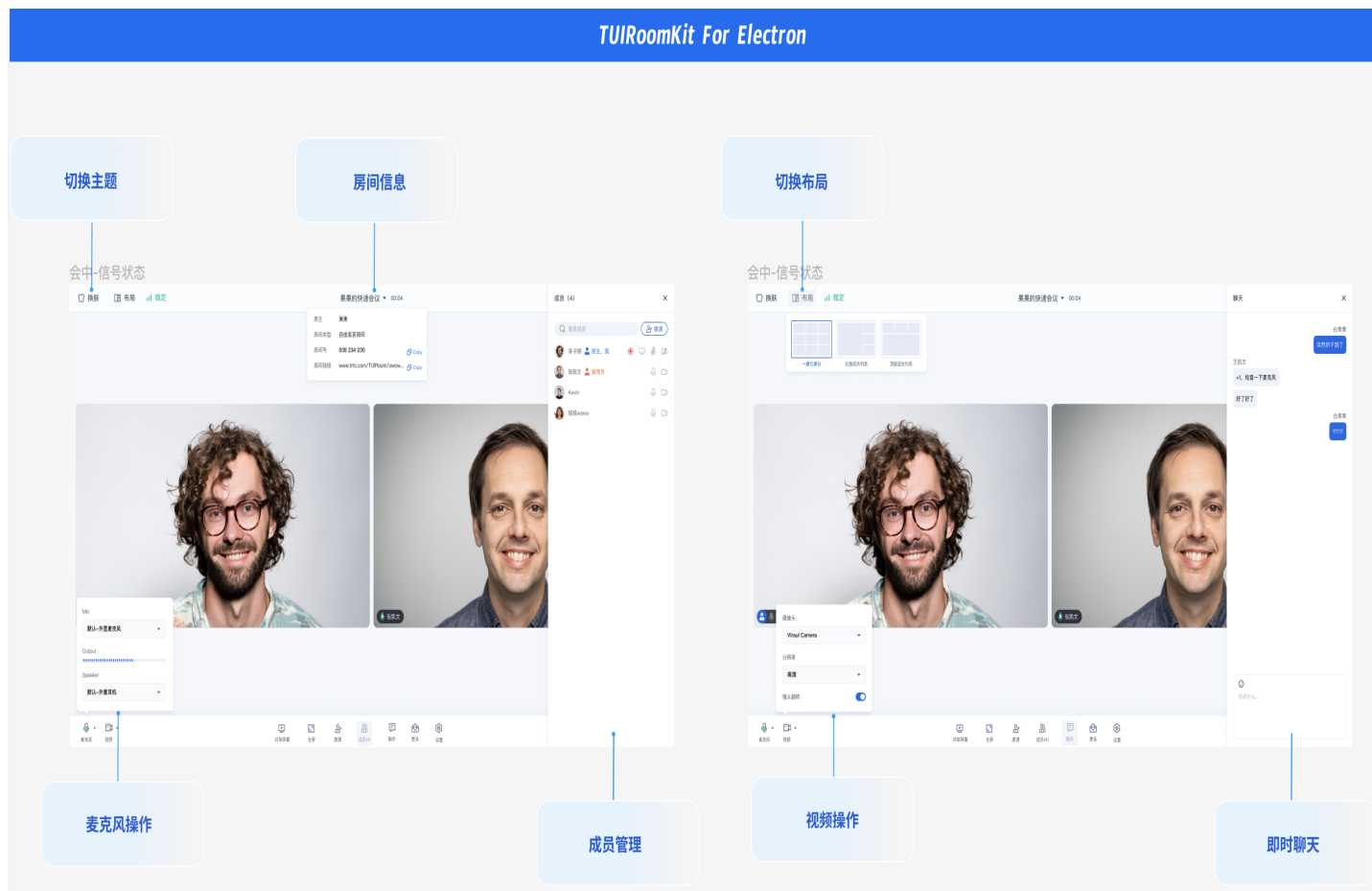
如果您在使用过程中有遇到什么问题，欢迎提交 [issue](#)，我们也欢迎您加入我们的 [腾讯云实时音视频 TRTC 技术交流群](#) 进行技术交流和反馈问题。

Electron (Windows&Mac)

最近更新时间：2024-11-06 09:38:02

TUIRoomKit 是腾讯云推出的多人音视频 UI 组件。组件提供房间管理，音视频控制，屏幕共享，成员管理，麦位管理，即时聊天，自定义布局切换等丰富的功能交互，同时支持中英文切换，一键换肤等能力。

本文介绍 TUIRoomKit (Electron) 的接入指引，助力您快速上线企业会议、在线教育、医疗问诊、在线巡视、远程定损等业务场景。



TUIRoomKit Demo 体验

您可以单击下载 [Mac OS版](#) 或 [Windows版](#) 体验 TUIRoomKit (Electron) 更多功能。

环境准备

- Node.js 版本: Node.js $\geq 16.19.1$ (推荐使用官方 LTS 版本, npm 版本请与 node 版本匹配)。
- Electron 版本: Electron $< 24.8.8$
- npm 包集成
 - Vue3 开发环境, 集成 [@tencentcloud/room-electron-vue3](#) NPM 包。

- Vue2.7 开发环境：集成 [@tencentcloud/roomkit-electron-vue2.7](#) NPM 包。

- 源码集成

- Vue3 + TypeScript 开发环境：从 [@tencentcloud/room-electron-vue3](#) NPM 包拷贝源码。
- Vue2.7 + TypeScript 开发环境：从 [@tencentcloud/roomkit-electron-vue2.7](#) NPM 包拷贝源码。

- 模板工程

- Electron + vite + vue3 [electron-vite-vue](#)

集成 TUIRoomKit 组件

如果您没有 Vue 项目，可以前往 [Github](#) 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Electron 示例工程。另外您如果需要在已有项目中进行集成，请按照以下步骤进行接入。

⚠ 注意：

若您通过下载模板工程 [electron-vite-vue](#) 的方式来进行集成，您需要切换模板工程至 v1.0.0，可参考如下指令：

```
git clone https://github.com/electron-vite/electron-vite-vue.git
cd electron-vite-vue
git checkout v1.0.0
```

步骤一：安装依赖

Vue3

```
npm install @tencentcloud/roomkit-electron-vue3 pinia@2.0.24 --save
```

Vue2

```
# 注意此处要求 vue版本 >= 2.7.16，如安装失败请检查您的 vue 版本是否支持
npm install @tencentcloud/roomkit-electron-vue2.7 pinia
```

步骤二：项目工程配置

1. 调整内容安全策略

在 Mac 系统中，默认的内容安全策略设置项会在加载 TUIRoomKit 组件时报错，为避免 TUIRoom 页面无法加载或者接口调用报错，可以调整内容安全策略。调整文件为 `packages/renderer/index.html` 文件。

```
<head>
  <meta charset="UTF-8" />
  <link rel="icon" href="/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- Mac下修改此设置 -->
  <meta http-equiv="Content-Security-Policy" content="script-src 'self' 'unsafe-inline' 'unsafe-eval'; worker-src 'self' blob:;"/>
  <title>Vite App</title>
</head>
```

2.注册pinia

TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia。项目入口文件为 `src/main.ts` 文件。

Vue3

```
// src/main.ts 文件
import { createPinia } from 'pinia';

const app = createApp(App);
// 注册 pinia
app.use(createPinia());
app.mount('#app')
```

Vue2

```
// src/main.ts 文件
import { createPinia, PiniaVuePlugin } from 'pinia';

Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
  pinia,
  render: h => h(App),
}).$mount('#app');
```

3.配置 vite.config.ts

为了统一代码风格并在 UI 层通过 import 方式引入 trtc-electron-sdk（否则必须通过 require 方式引入），您需要在 `packages/renderer/vite.config.ts` 中进行配置。请参照以下配置项替换 resolve 中的内容，具体可参考文件 [packages/renderer/vite.config.ts](#)。

⚠ 注意：

若您的工程采用的是 vite 打包方式，请按如下步骤进行配置。原因是 vite 默认只支持 ES6 模块，而 trtc-electron-sdk 需要对接 Node.js API，其是一个 Common JS 模块，此步骤的目的是为了进行模块类型的兼容。若您的工程采用的是 webpack 打包方式，则可跳过配置 vite.config.ts 这一步。

```
// vite.config.ts
export default defineConfig({
  // ...
  plugins: [
    resolve({
      'trtc-electron-sdk': `
        const TRTCCloud = require("trtc-electron-sdk");
        const TRTCParams = TRTCCloud.TRTCParams;
        const TRTCAppScene = TRTCCloud.TRTCAppScene;
        const TRTCVideoStreamType = TRTCCloud.TRTCVideoStreamType;
        const TRTCScreenCaptureSourceType =
TRTCCloud.TRTCScreenCaptureSourceType;
        const TRTCVideoEncParam = TRTCCloud.TRTCVideoEncParam;
        const Rect = TRTCCloud.Rect;
        const TRTCAudioQuality = TRTCCloud.TRTCAudioQuality;
        const TRTCScreenCaptureSourceInfo =
TRTCCloud.TRTCScreenCaptureSourceInfo;
        const TRTCDeviceInfo = TRTCCloud.TRTCDeviceInfo;
        const TRTCVideoQosPreference =
TRTCCloud.TRTCVideoQosPreference;
        const TRTCQualityInfo = TRTCCloud.TRTCQualityInfo;
        const TRTCQuality = TRTCCloud.TRTCQuality;
        const TRTCStatistics = TRTCCloud.TRTCStatistics;
        const TRTCVolumeInfo = TRTCCloud.TRTCVolumeInfo;
        const TRTCDeviceType = TRTCCloud.TRTCDeviceType;
        const TRTCDeviceState = TRTCCloud.TRTCDeviceState;
        const TRTCBeautyStyle = TRTCCloud.TRTCBeautyStyle;
        const TRTCVideoResolution = TRTCCloud.TRTCVideoResolution;
        const TRTCVideoResolutionMode =
TRTCCloud.TRTCVideoResolutionMode;
        const TRTCVideoMirrorType = TRTCCloud.TRTCVideoMirrorType;
        const TRTCVideoRotation = TRTCCloud.TRTCVideoRotation;
        const TRTCVideoFillMode = TRTCCloud.TRTCVideoFillMode;
```

```
const TRTCRoleType = TRTCCloud.TRTCRoleType;
const TRTCScreenCaptureProperty =
TRTCCloud.TRTCScreenCaptureProperty;
export {
  TRTCParams,
  TRTCAppScene,
  TRTCVideoStreamType,
  TRTCScreenCaptureSourceType,
  TRTCVideoEncParam,
  Rect,
  TRTCAudioQuality,
  TRTCScreenCaptureSourceInfo,
  TRTCDeviceInfo,
  TRTCVideoQosPreference,
  TRTCQualityInfo,
  TRTCStatistics,
  TRTCVolumeInfo,
  TRTCDeviceType,
  TRTCDeviceState,
  TRTCBeautyStyle,
  TRTCVideoResolution,
  TRTCVideoResolutionMode,
  TRTCVideoMirrorType,
  TRTCVideoRotation,
  TRTCVideoFillMode,
  TRTCRoleType,
  TRTCQuality,
  TRTCScreenCaptureProperty,
};
export default TRTCCloud.default;
},
]),
];
});
```

步骤三：引用 TUIRoomKit 组件

❗ 说明：

引入 ConferenceMainView 组件，组件默认处于 **常驻模式**（组件始终显示，内部不控制组件的显示与隐藏，若业务端不进行控制，则组件一直保持显示状态）。

Vue3

```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script setup>
import { ConferenceMainView } from '@tencentcloud/roomkit-electron-vue3';
</script>
```

Vue2

```
<template>
  <ConferenceMainView></ConferenceMainView>
</template>
<script>
import { ConferenceMainView } from '@tencentcloud/roomkit-electron-vue2.7';
export default {
  components: {
    ConferenceMainView,
  },
};
</script>
```

步骤四：登录 TUIRoomKit 组件

开启会议前需要调用 [login](#) 接口进行登录。获取 sdkAppId、userId、userSig 可参见 [开通服务](#)。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.login({
  sdkAppId: 0, // 获取 sdkAppId 可参考文档开通服务部分，
https://cloud.tencent.com/document/product/647/104842
  userId: '', // 用户在您业务中的唯一标示 Id
  userSig: '', // 本地开发调试可在
https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig，注意 userSig 与 userId 为一一对应关系
});
```

参数	类型	说明
userID	String	客户根据自己的业务自定义用户 ID，只允许包含大小写英文字母(a-z A-Z)、数字(0-9)及下划线和连词符。
sdkAppID	int	在 实时音视频 TRTC 控制台 创建的音视频应用的唯一标识 SDKAppID。
secretKey	String	在 实时音视频 TRTC 控制台 创建的音视频应用的 SDKSecretKey。
userSig	String	一种安全保护签名，用于对用户进行登录鉴权认证，确认用户是否真实，阻止恶意攻击者盗用您的云服务使用权。

❗ 说明：

- **开发环境：**如果您正在本地开发调试阶段，可以采用本地 `GenerateTestUserSig.genTestSig` 函数生成 userSig。该方法中 SDKSecretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量。
- **生产环境：**如果您的项目要发布上线，请采用 [服务端生成 UserSig](#) 的方式。

步骤五：发起新的会议

会议主持人可以通过调用 [start](#) 接口来发起一场新的会议，其他参会议者可以参见 [步骤六](#) 的描述，调用 [join](#) 接口加入该会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
const startConference = async () => {
  await conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
  await conference.start('123456', {
    roomName: 'TestRoom',
    isSeatEnabled: false,
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
}
startConference()
```

步骤六：进入已有会议

参与者可以通过调用 `join` 接口，填写对应的 `roomId` 参数，来加入由会议主持人在 [步骤五](#) 中发起的会议。

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
const joinConference = async () => {
  await conference.login({
    sdkAppId: 0, // Replace with your sdkAppId
    userId: '', // Replace with your userId
    userSig: '', // Replace with your userSig
  });
  await conference.join('123456', {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
}
joinConference()
```

开发环境运行

1. 执行开发环境命令。（此处以 `vue3 + vite` 默认项目为例，不同项目 `dev` 指令可能不同，请根据您的项目进行调整）

```
npm run dev
```

⚠ 注意：

运行过程中若 `src/TUIRoom` 目录中有 `eslint` 报错，可在 `.eslintignore` 文件中添加 `/src/TUIRoom/` 路径屏蔽 `eslint` 检查。

2. 体验 `TUIRoomKit` 组件功能。

生产环境部署

项目进行打包

```
npm run build
```

ⓘ 说明：

以模板工程接入为例，执行上述打包命令之后，会在您的项目根目录下生成 **release** 文件夹，其中包含您所需要的对应安装包。

常见问题

若您在步骤一安装依赖时出现 vue 版本不匹配等问题，如何解决？

需要在 package.json 文件中将 vue 版本固定为 3.3.13

如果您在运行阶段产生了 app crash，如何解决？

原因可能是获取不到摄像头和麦克风的权限而引起的。

解决方案：

1. 请检查您的摄像头或麦克风设备使用是否正常，有无被其他应用占用。
2. 在 /packages/main/index.ts 中，注释掉 `checkAndApplyDevicePrivilege` 方法的调用，即可跳过设备权限的检查。

若您在运行阶段或打包阶段遇到如下图报错，如何解决？

```
error TS6504: File 'D:/electron-vite-vue/packages/renderer/src/App.vue.__VLS_template.jsx' is a JavaScript file. Did you mean to enable the 'allowJs' option?
The file is in the program because:
  Root file specified for compilation

error TS6504: File 'D:/electron-vite-vue/packages/renderer/src/App.vue.js' is a JavaScript file. Did you mean to enable the 'allowJs' option?
The file is in the program because:
  Root file specified for compilation
```

需要在 tsconfig.json 文件中启用 allowJs 选项：

```
// tsconfig.json
{
  "compilerOptions": {
    "allowJs": true
    // ...其他选项
  }
  // ...其他配置
}
```

若您在运行或打包阶段遇到如下图报错，如何解决？

```
✖ Uncaught ReferenceError: exports is not defined                                index.a12c1be5.js:1
   at index.a12c1be5.js:1:706
```

需要您在 index.html 文件中进行如下增加如下内容：

```
// index.html
<head>
  ....
  <script>const exports = module.exports;</script>
</head>
```

其他文档

- [TUIRoomKit](#)
- [TUIRoom Demo 快速跑通](#)
- [设置昵称、头像](#)
- [预定会议](#)
- [界面定制 \(TUIRoomKit\)](#)
- [常见问题](#)

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 [腾讯云实时音视频 TRTC 技术交流群](#) 进行技术交流和问题反馈。