音视频通话 SDK

快速接入

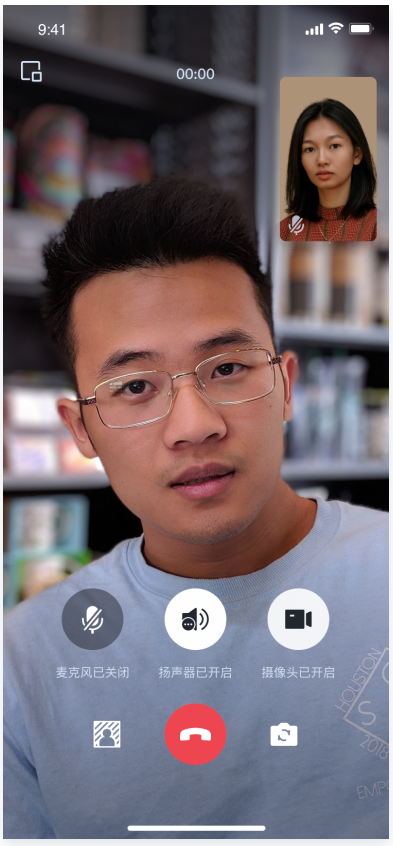
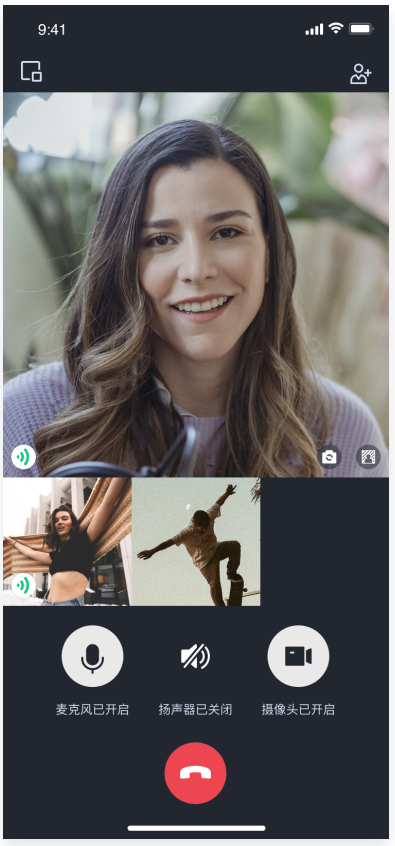
腾讯云

文档目录

快速接入
Android

最近更新时间：2025-06-10 09:37:01

本文将介绍如何快速完成 TUICallKit 组件的接入，您将在 10 分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

1v1 视频通话	群组通话
	

环境准备

- Android 5.0（SDK API Level 21）及以上版本。
- Gradle 4.2.1 及以上的版本。
- Android 5.0 及以上的手机设备。

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。具体步骤详见 开通服务 。开通服务后，请记录 `SDKAppID` 和 `SDKSecretKey`，在后续的步骤中会用到。

步骤二：下载并导入组件

在 Github 中克隆/下载代码，然后拷贝 Android 目录下的 tuicallkit-kt 子目录到您当前工程中的 app 同一级目录中，如下图所示。

步骤三：工程配置

1. 在工程根目录下找到 `settings.gradle.kts`（或`settings.gradle`）文件，在其中增加如下代码，导入 `tuicallkit-kt` 组件到项目中。

setting.gradle.kts

```
include(":tuicallkit-kt")
```

settings.gradle

```
include ':tuicallkit-kt'
```

2. 在 app 目录下找到 `build.gradle.kts`(或`build.gradle`) 文件，在 `dependencies` 中增加如下代码，声明当前 app 对新加入的组件的依赖。

build.gradle.kts

```
dependencies {
    api(project(":tuicallkit-kt"))
}
```

build.gradle

```
dependencies {
    api project(':tuicallkit-kt')
}
```

> **说明**
> TUICallKit 工程内部已经默认依赖：`TRTC SDK`、`IM SDK`、`tuicallengine` 以及公共库 `tuicore`，不需要开发者单独配置。如需进行版本升级，则修改 `tuicallkit-kt/build.gradle` 文件中的版本号即可。

3. 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 app 目录下的 `proguard-rules.pro` 文件末尾添加如下代码。添加完后，点击右上角的“Sync Now”，同步代码。

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

4. 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">
```

5. 建议您编译并运行一次。如果遇到问题，建议您尝试运行我们的 Github demo 项目。通过比对，您可以找出潜在的区别并解决遇到的问题。在接入和使用过程中，如果遇到问题，欢迎向我们反馈 。

步骤四：登录 TUI 组件

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 组件的登录。这一步骤至关重要，只有在成功登录之后，您才能正常使用 TUICallKit 提供的各项功能。

Kotlin

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

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // begin
        val userID = "denny"      // 请替换为您的 UserId
        val sdkAppID = 0          // 请替换为第一步在控制台得到的SDKAppID
        val secretKey = "****"    // 请替换为第一步在控制台得到的SecretKey

        val userSig = GenerateTestUserSig.genTestUserSig(userId, sdkAppId, secretKey)

        TUILogin.login(this, sdkAppId, userId, userSig, object : TUICallback() {
            override fun onSuccess() {
            }
            override fun onError(errorCode: Int, errorMessage: String) {
            }
         })
        // end
    }
```

```
}
```

Java

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

public class MainActivity extends AppCompatActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //begin
        String userID = "denny";      // 请替换为您的UserId
        int sdkAppID = 0;             // 请替换为第一步在控制台得到的SDKAppID
        String secretKey = "****";    // 请替换为第一步在控制台得到的SecretKey

        String userSig = GenerateTestUserSig.genTestUserSig(userId, sdkAppId, secretKey);

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

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

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：拨打您的第一通电话

在上述登录方法调用返回成功后，调用TUICallKit 的 call 方法，指定被叫方的 userID 和通话类型，发起音视频通话，被叫方可接收到来电邀请。

Kotlin

```
import com.tencent.qcloud.tuikit.tuicallengine.TUICallDefine
import com.tencent.qcloud.tuikit.tuicallkit.TUICallKit

// 发起1对1语音通话（假设被叫方的 userID 为 mike）
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Audio)
```

Java

```
import com.tencent.qcloud.tuikit.tuicallengine.TUICallDefine;
import com.tencent.qcloud.tuikit.tuicallkit.TUICallKit;

// 发起1对1语音通话（假设被叫方的 userID 为 mike）
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Audio);
```

主叫方发起音频通话	被叫方收到音频通话请求
	

更多特性

- 设置昵称、头像
- 界面定制
- 离线推送
- 群组通话
- 悬浮窗
- 美颜特效
- 自定义铃声
- 监听通话状态
- 云端录制

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 。

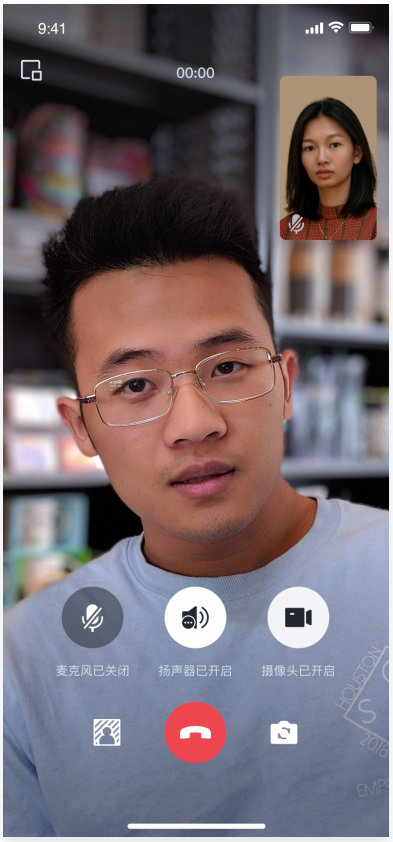
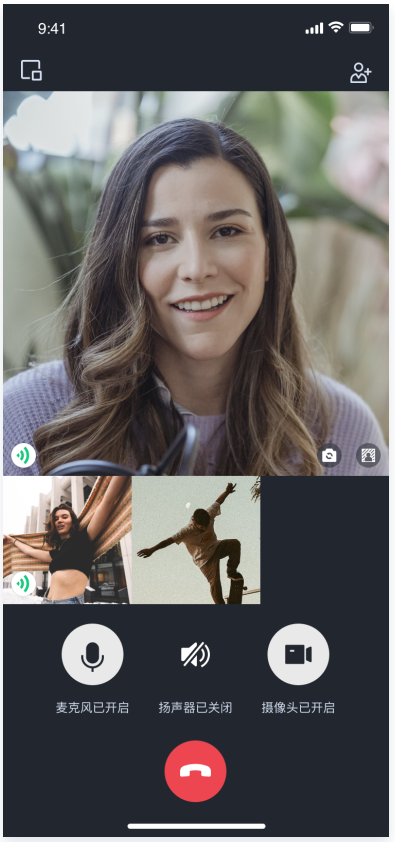
交流与反馈

- 如果您在使用过程中，有什么建议或者意见，可以在这里反馈：TUICallKit 产品反馈问卷 ，感谢您的反馈。
- 如果您是开发者，也欢迎您加入我们的 TUICallKit 技术交流平台 zhiliao ，进行技术交流和产品沟通。

iOS

最近更新时间：2025-06-09 11:21:42

本文将介绍如何快速完成 TUICallKit 组件的接入，您将在 10 分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

1v1 视频通话	群组通话
	

环境准备

- Xcode 13 及以上。
- iOS 13.0 及以上。
- CocoaPods 环境安装，单击 查看 。
- 如果您的接入和使用中遇到问题，请参见 常见问题 。

步骤1：开通服务

请参见 开通服务 ，获取 `SDKAppID`、`SDKSecretKey` ，他们将在 步骤4：初始化 TUICallKit 组件 作为**必填参数**使用。

步骤2：导入组件

使用 CocoaPods 导入组件，如果您遇到问题，请先参见 环境准备 。导入组件具体步骤如下：

1. 请在您的 `Podfile` 文件中添加 `pod 'TUICallKit_Swift'` 依赖，建议指定 `Subspec` 为 `Professional` ，如果您遇到任何问题，请参见 Example 工程。

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

```
  pod 'TUICallKit_Swift/Professional'
end
```

说明：
如果您的项目中缺少 `Podfile` 文件，您需要在终端中 `cd` 到 `xxxx.xcodeproj` 目录，然后，通过执行以下命令来创建 `Podfile` 文件：

```
pod init
```

2. 在终端中，首先 `cd` 到 `Podfile` 目录下，然后执行以下命令，安装组件。

```
pod install
```

说明：
如果无法安装 TUICallKit 最新版本，可以先删除 **Podfile.lock** 和 **Pods**。然后执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

之后执行以下命令，更新组件库的 Pod 版本。

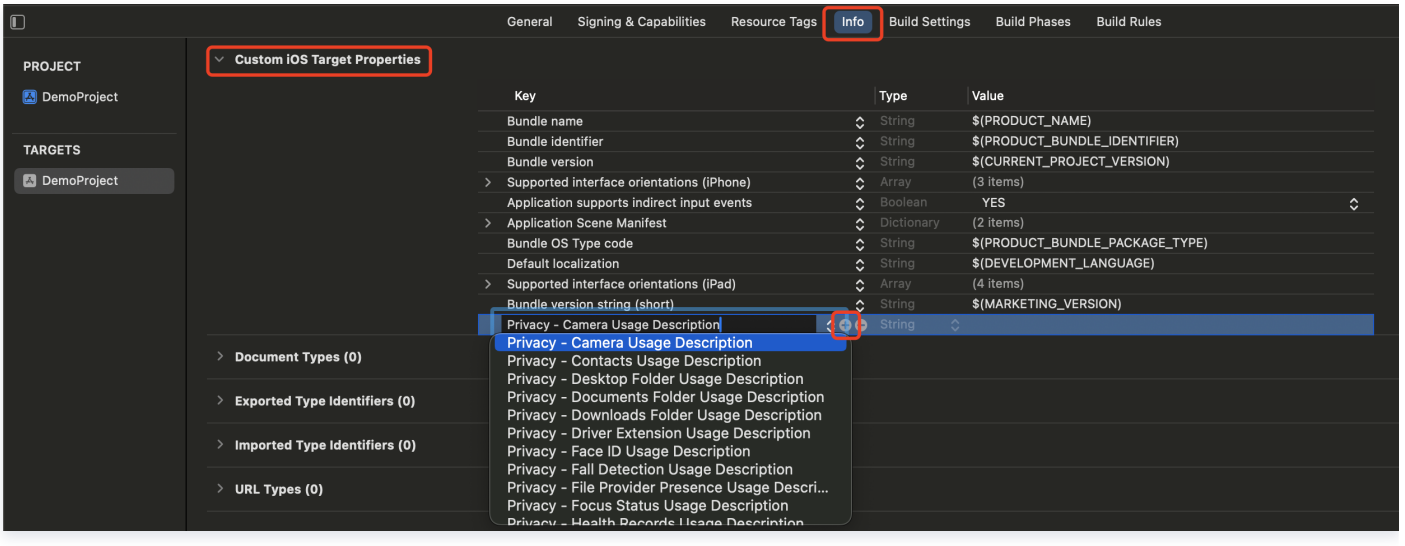
```
pod update
```

3. 建议您编译并运行一次。如果遇到问题，可以参见我们的 常见问题 。如果问题仍未解决，您可以尝试运行我们的 Example 工程。在接入和使用过程中，如果遇到问题，欢迎向我们 反馈 。

步骤3：工程配置

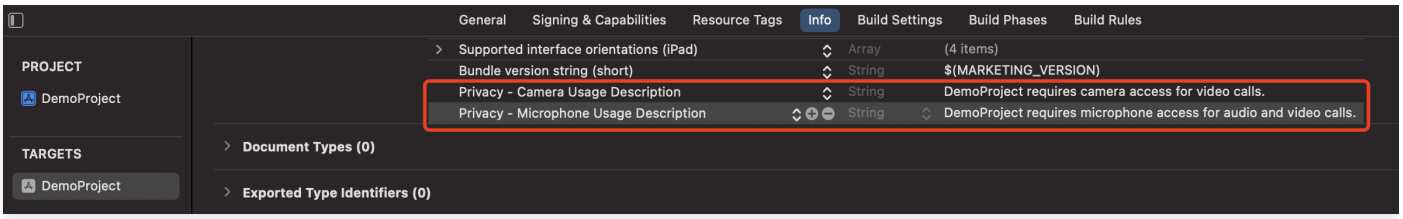
使用音视频功能，需要授权摄像头和麦克风的使用权限，请根据实际项目需要，设置项目所需权限。

1. 在 Xcode 中，选择 TARGETS > Info > Custom iOS Target Properties 菜单。



2. 单击 +，添加摄像头和麦克风权限。

- `Privacy - Camera Usage Description`
- `Privacy - Microphone Usage Description`



步骤4：登录 TUI 组件

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 组件的登录。这一步骤至关重要，只有在成功登录之后，您才能正常使用 TUICallKit 提供的各项功能。

Swift

```
import TUICore
import TUICallKit_Swift

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {

    let userID = "denny"        // 请替换为您的 UserId
    let sdkAppID: Int32 = 0     // 请替换为第一步在控制台得到的 SDKAppID
    let secretKey = "****"      // 请替换为第一步在控制台得到的 SecretKey

    let userSig = GenerateTestUserSig.genTestUserSig(userID: userID, sdkAppID: sdkAppID,
secretKey: secretKey)

    TUILogin.login(sdkAppID, userID: userID, userSig: userSig) {
      print("login success")
    } fail: { code, message in
      print("login failed, code: \(code), error: \(message ?? "nil")")
    }

    return true
}
```

Objective-C

```
#import <TUICore/TUILogin.h>
#import <TUICallKit_Swift/TUICallKit_Swift-Swift.h>

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {

    NSString *userID = @"denny";      // 请替换为您的 UserId
    int sdkAppID = 0;                 // 请替换为第一步在控制台得到的 SDKAppID
    NSString *secretKey = @"****";    // 请替换为第一步在控制台得到的 SecretKey
```

```
    NSString *userSig = [GenerateTestUserSig genTestUserSigWithUserID:userID sdkAppID:sdkAppID
secretKey:secretKey];

    [TUILogin login:sdkAppID
             userID:userID
            userSig:userSig
               succ:^{
        NSLog(@"login success");
    } fail:^(int code, NSString * _Nullable msg) {
        NSLog(@"login failed, code: %d, error: %@", code, msg);
    }];

    return YES;
}
```

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	Int32	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。

⚠ 注意：

- **开发环境：**如果您处于本地开发调试阶段，可以使用本地 GenerateTestUserSig.genTestSig 函数生成 userSig。但请注意，该方法中的 SDKSecretKey 容易被反编译逆向破解。一旦密钥泄露，攻击者可能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5：拨打您的第一通电话

通过调用 TUICallKit 的 call 函数并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话。

Swift

```
import TUICallKit_Swift
import TUICallEngine

// 发起1对1语音通话（假设 userId 为 mike）
TUICallKit.createInstance().call(userId: "mike", callMediaType: .audio)
```

Objective-C

```
#import <TUICallKit_Swift/TUICallKit_Swift-Swift.h>
#import <TUICallEngine/TUICallEngine.h>

// 发起1对1语音通话（假设 userId 为 mike）
[[TUICallKit createInstance] callWithUserId:@"mike" callMediaType:TUICallMediaTypeAudio];
```

主叫方	被叫方
	

更多特性

- 设置昵称、头像
- 界面定制
- 离线推送
- 群组通话
- 悬浮窗
- 美颜特效
- 自定义铃声
- 监听通话状态
- 云端录制

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 。

交流与反馈

- 如果您在使用过程中，有什么建议或者意见，可以在这里反馈： TUICallKit 产品反馈问卷 ，感谢您的反馈。
- 如果您是开发者，也欢迎您加入我们的 TUICallKit 技术交流平台 zhiliao ，进行技术交流和产品沟通。

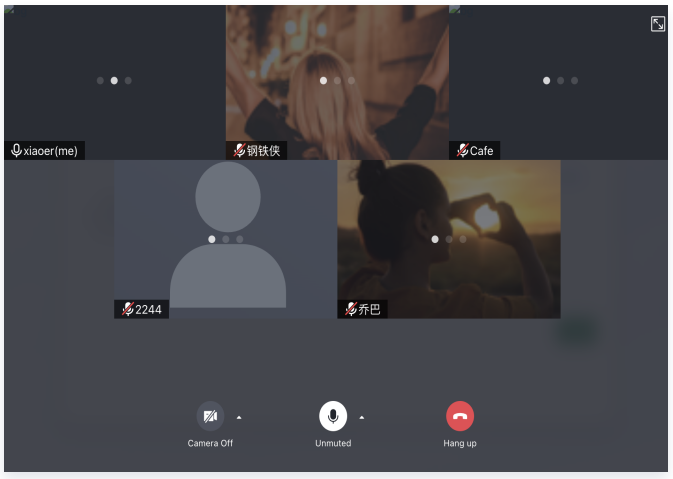
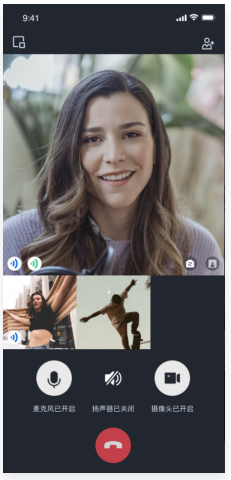
Web&H5(Vue2/Vue3)

最近更新时间：2025-05-16 17:17:42

本文介绍群组通话功能的使用，如发起群组通话、加入群组通话。

预期效果

TUICallKit 支持群组通话，预期效果见下图。

Web 端	移动端
	

发起多人通话

调用 `calls` API 发起群通话。

```
try {
   const params = {
     userIDList: ['user1', 'user2'],
     type: TUICallType.VIDEO_CALL,
   }
  await TUICallKitServer.calls(params);
} catch (error: any) {
  alert(`[TUICallKit] groupCall failed. Reason:${error}`);
}
```

加入通话

调用 join API 主动加入已有的音视频通话。

```
try {
  await TUICallKitServer.join({callId: 'xxx'});
} catch (error: any) {
  alert(`[TUICallKit] join failed. Reason: ${error}`);
}
```

uni-app（小程序）

最近更新时间：2025-05-16 17:17:42

本文将介绍如何快速完成 TUICallKit 组件的接入，跟随本文档，您将在半小时内得到一个包含完备 UI 界面的视频通话小程序。基本功能如下图所示：

> **注意：**
> cli 脚手架工具创建的项目集成，集成指引请点击 uni-app（小程序）脚手架 查看。

小程序 Demo 体验

- 如果您想要直接体验音视频通话小程序，单击 Demo 体验，扫描小程序二维码。
- 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** TUICallKit 组件，搭建一个音视频通话小程序，请跟随本文档。

开发环境要求

- 微信 App iOS 最低版本要求：8.0.40。
- 微信 App Android 最低版本要求：8.0.40。
- 小程序基础库最低版本要求：2.10.0。

> **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。

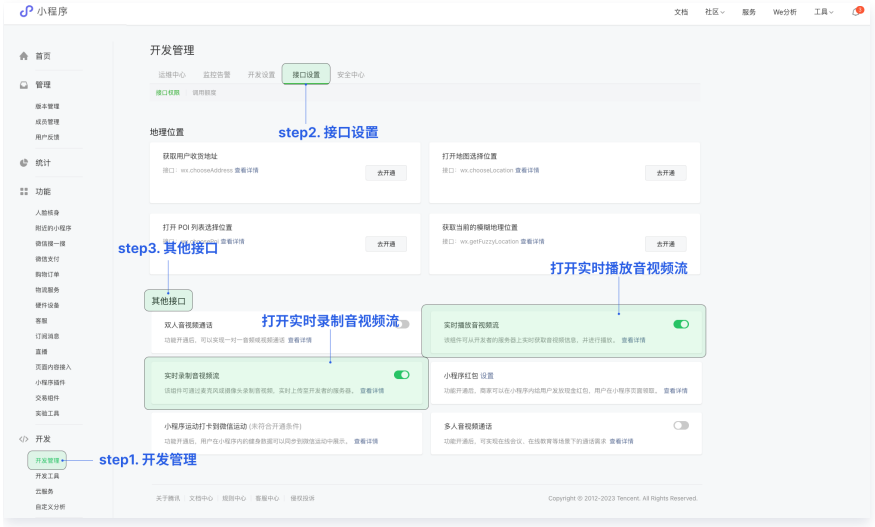
小程序开发准备

步骤一：开通企业类小程序

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：

步骤二：在小程序控制台开启实时音视频接口

- 小程序推拉流标签使用权限暂时只开放给有限类目，具体支持类目参见该地址 。
- 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。



步骤三：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。

- 将以下域名添加到 socket 合法域名：

域名	说明	是否必须
`wss://${SDKAppID}w4c.my-imcloud.com`	v3.4.6起，SDK 支持独立域名，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`	必须
`wss://wss.im.qcloud.com`	Web IM 业务域名	必须
`wss://wss.tim.qq.com`	Web IM 业务域名	必须
`wss://wssv6.im.qcloud.com`	Web IM 业务域名	必须

- 将以下域名添加到 request 合法域名：

域名	说明	是否必须
https://web.sdk.qcloud.com	Web IM 业务域名	必须
https://boce-cdn.my-imcloud.com	Web IM 业务域名	必须
https://api.im.qcloud.com	Web IM 业务域名	必须
https://events.im.qcloud.com	Web IM 业务域名	必须
https://webim.tim.qq.com	Web IM 业务域名	必须
https://wss.im.qcloud.com	Web IM 业务域名	必须
https://wss.tim.qq.com	Web IM 业务域名	必须

- 将以下域名添加到 uploadFile 合法域名：

域名	说明	是否必须
https://${SDKAppID}-cn.rich.my-imcloud.com	从 2024年9月10日起，新增应用默认分配 cos 独立域名。例如您的 SDKAppID 是 1400xxxxxx，则 cos 独立域名为：https://1400xxxxxx-cn.rich.my-imcloud.com	必须
https://cn.rich.my-imcloud.com	文件上传域名	必须
https://cn.imrich.qcloud.com	文件上传域名	必须
https://cos.ap-shanghai.myqcloud.com	文件上传域名	必须
https://cos.ap-shanghai.tencentcos.cn	文件上传域名	必须
https://cos.ap-guangzhou.myqcloud.com	文件上传域名	必须

- 将以下域名添加到 downloadFile 合法域名：

域名	说明	是否必须
https://${SDKAppID}-cn.rich.my-imcloud.com	从 2024年9月10日起，新增应用默认分配 cos 独立域名。例如您的 SDKAppID 是 1400xxxxxx，则 cos 独立域名为：https://1400xxxxxx-cn.rich.my-imcloud.com	必须
https://cn.rich.my-imcloud.com	文件下载域名	必须
https://cn.imrich.qcloud.com	文件下载域名	必须
https://cos.ap-shanghai.myqcloud.com	文件下载域名	必须
https://cos.ap-shanghai.tencentcos.cn	文件下载域名	必须

`https://cos.ap-guangzhou.myqcloud.com`	文件下载域名	必须

- 如下图所示，服务器域名配置：

TUICallKit 源码集成

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 小程序项目

1. 在 HBuilder 中创建小程序项目。

2. 在终端输入 `npm init -y` ，创建 `package.json` 文件。

```
npm init -y
```

步骤三：下载并导入 TUICallKit 组件

MacOS 端

```
npm i @tencentcloud/call-uikit-wx-uniapp
```

```
mkdir -p ./TUICallKit && cp -r node_modules/@tencentcloud/call-uikit-wx-uniapp/
./TUICallKit && cp node_modules/@tencentcloud/call-engine-wx/RTCCallEngine.wasm.br
./static
```

Windows 端

```
npm i @tencentcloud/call-uikit-wx-uniapp
```

```
xcopy node_modules\@tencentcloud\call-uikit-wx-uniapp\ .\TUICallKit /i /e
xcopy node_modules\@tencentcloud\call-engine-wx\RTCCallEngine.wasm.br .\static
```

步骤四：使用精简版 IM SDK 减小体积（可选）

> **警告：**
> 精简版 IM SDK 体积降低 **550KB**，仅适用于独立集成 TUICallKit 的场景。

MacOS 端

```
npm install @tencentcloud/lite-chat && rm -rf node_modules/@tencentcloud/chat && cp -r
node_modules/@tencentcloud/lite-chat node_modules/@tencentcloud/chat
```

Windows 端

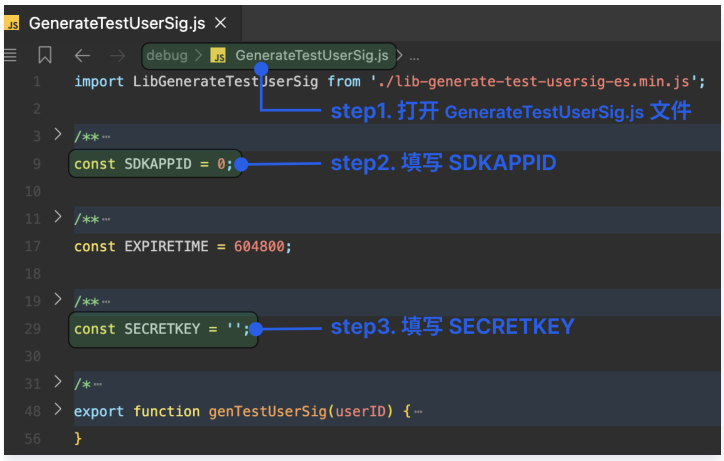
```
npm install @tencentcloud/lite-chat
```

```
rmdir /s /q node_modules\@tencentcloud\chat
```

```
xcopy node_modules\@tencentcloud\lite-chat node_modules\@tencentcloud\chat /i /e
```

步骤五：填写 SDKAPPID 以及 SECRETKEY

修改 TUICallKit/debug/ `GenerateTestUserSig-es.js` 文件 的 SDKAPPID 以及 SECRETKEY。



步骤六：调用 TUICallKit 组件

说明：

下面会重点介绍各种接入方式包体积占比，请针对自身项目，选择合适的接入方式。

接入方式	主包接入	分包接入
能否使用全局监听	是	否
是否占用主包体积	是	否
Vue2 项目体积占比	约占用 1.3MB 主包体积，其中 TUICallKit/debug 目录用于生成 userSig，生产环境推荐在服务端生成，可以节省约 150kb 主包体积。	不会占用任何主包体积。
Vue3 项目体积占比	约占用 1.3MB 主包体积，其中 TUICallKit/debug 目录用于生成 userSig，生产环境推荐在服务端生成，可以节省约 150kb 主包体积。	参见 体积优化文档，不会占用任何主包体积。

全局监听：使小程序具有所有页面都能唤起通话页面的能力。

主包接入

注意：

主包接入的方式默认集成全局监听。

在 pages.json 文件注册全局监听页面。

```
    {
```

```
        "path": "TUICallKit/src/Components/TUICallKit",
        "style": {
            "navigationBarTitleText": "uni-app"
        }
    }
```

Vue3

1. 修改 **pages/index/index.vue** 文件。

```
<template>
  <view class="loginBox">
    <input
      class="input-box"
      v-model="userID"
      :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
      placeholder-style="color:#BBBBBB;"
    />
    <view class="login">
      <button
        class="loginBtn"
        @click="!isLogin ? loginHandler() : callHandler()"
      >
        {{ !isLogin ? "登录" : "呼叫" }}
      </button>
    </view>
  </view>
</template>
<script setup>
import { ref } from "vue";
import * as GenerateTestUserSig from "../../TUICallKit/debug/GenerateTestUserSig-
es.js";
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/src/index";
// 导入 CallManager 模块，使您的应用具有全局监听来电的能力
import { CallManager } from "../../TUICallKit/src/TUICallService/serve/callManager";
uni.CallManager = new CallManager();
let userID = ref("");
let isLogin = ref(false);
const loginHandler = async () => {
  if (!userID.value) return;
  const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
    userID: userID.value,
  });
  await uni.CallManager.init({
    sdkAppID: SDKAppID,
    userID: userID.value,
    userSig: userSig,
    globalCallPagePath: "TUICallKit/src/Components/TUICallKit",
  });
  isLogin.value = true;
  userID.value = "";
};
```

```
const callHandler = async () => {
  await TUICallKitServer.calls({
    userIDList: [userID.value],
    type: 2,
  });
};
</script>
<style>
.loginBox {
  margin-top: 200px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

input {
  display: flex;
  font-size: 20px;
}

.login {
  width: 100vw;
  bottom: 5vh;
  margin: 70rpx;
}

.login button {
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
</style>
```

Vue2

1. **下载依赖。**

```
npm i unplugin-vue2-script-setup
```

2. **修改 `vue.config.js` 文件。**

```
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;
module.exports = {
  parallel: false,
  configureWebpack: {
    plugins: [
      ScriptSetup({
        /* options */
```

```
      }),
    ],
  },
  chainWebpack(config) {
    config.plugins.delete('fork-ts-checker');
  },
};
```

3. 修改 `main.js` 文件。

```
 import vueComposition from "@vue/composition-api"
 Vue.use(vueComposition)
```

4. 修改 **pages/index/index.vue** 文件。

```
<template>
<view>
  <view class="loginBox">
    <input
      class="input-box"
      v-model="userID"
      :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
      placeholder-style="color:#BBBBBB;"
    />
    <view class="login">
      <button
        class="loginBtn"
        @click="!isLogin ? loginHandler() : callHandler()"
      >
        {{ !isLogin ? "登录" : "呼叫" }}
      </button>
    </view>
  </view>
</view>
</template>
<script>
import * as GenerateTestUserSig from "../../TUICallKit/debug/GenerateTestUserSig-
es.js";
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/src/index";
// 导入 CallManager 模块，使您的应用具有全局监听来电的能力
import { CallManager } from "../../TUICallKit/src/TUICallService/serve/callManager";
uni.CallManager = new CallManager();
export default {
  data() {
    return {
      isLogin: false,
      userID: "",
    };
  },
  methods: {
    async loginHandler() {
      if (!this.userID) return;
```

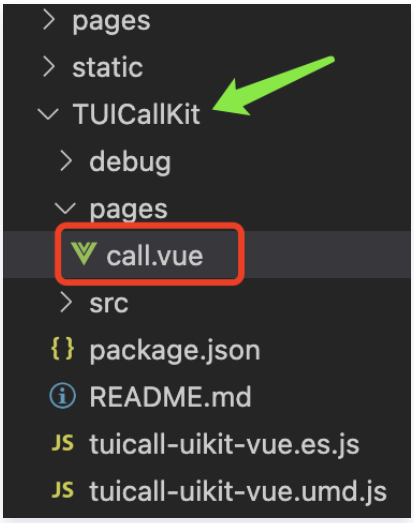
```
      const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
        userID: this.userID,
      });
      await uni.CallManager.init({
        sdkAppID: SDKAppID,
        userID: this.userID,
        userSig: userSig,
        globalCallPagePath: "TUICallKit/src/Components/TUICallKit",
      });
      this.isLogin = true;
      this.userID = "";
    },

    async callHandler() {
      await TUICallKitServer.calls({
        userIDList: [this.userID],
        type: 2,
      });
    },
  },
};
</script>
<style>
.loginBox {
  margin-top: 200px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

input {
  display: flex;
  font-size: 20px;
}

.login {
  width: 100vw;
  bottom: 5vh;
  margin: 70rpx;
}

.login button {
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
</style>
```

分包接入

1. 修改 manifest.json 小程序相关配置。

```
"mp-weixin": {
     "usingComponents": true,
      "optimization": {
         "subPackages": true
      }
 },
```

2. 新建 /TUICallkit/pages/call.vue 目录。



3. 修改 pages.json，添加分包页面。

```
"subPackages": [
      {
        "root": "TUICallKit",
        "pages": [
          {
            "path": "pages/call",
            "style": {
              "navigationBarTitleText": "uni-app"
            }
          }
        ]
      }
],
```

4. 集成 TUICallKit。

Vue3 环境

修改 TUICallKit/pages/call.vue 文件。

```
<template>
  <view>
```

```
    <TUICallKit></TUICallKit>
    <view class="loginBox">
      <input
        class="input-box"
        v-model="userID"
        :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
        placeholder-style="color:#BBBBBB;"
      />
      <view class="login">
        <button
          class="loginBtn"
          @click="!isLogin ? loginHandler() : callHandler()"
        >
          {{ !isLogin ? "登录" : "呼叫" }}
        </button>
      </view>
    </view>
  </view>
</template>
<script setup>
import { ref } from "vue";
import * as GenerateTestUserSig from "../debug/GenerateTestUserSig-es.js";
import { TUICallKitServer } from "../src/index";
import TUICallKit from "../src/Components/TUICallKit";
let userID = ref("");
let isLogin = ref(false);
const loginHandler = async () => {
  if (!userID.value) return;
  const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
    userID: userID.value,
  });
  await TUICallKitServer.init({
    sdkAppID: SDKAppID,
    userID: userID.value,
    userSig: userSig,
  });
  isLogin.value = true;
  userID.value = "";
};

const callHandler = async () => {
  await TUICallKitServer.calls({
    userIDList: [userID.value],
    type: 2,
  });
};
</script>
<style>
.loginBox {
  margin-top: 200px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}
```

```
input {
  display: flex;
  font-size: 20px;
}

.login {
  width: 100vw;
  bottom: 5vh;
  margin: 70rpx;
}

.login button {
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
</style>
```

Vue2 环境

1. 下载依赖。

```
npm i unplugin-vue2-script-setup
```

2. 修改 `vue.config.js` 文件。

```
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;
module.exports = {
  parallel: false,
  configureWebpack: {
    plugins: [
      ScriptSetup({
        /* options */
      }),
    ],
  },
  chainWebpack(config) {
    config.plugins.delete('fork-ts-checker');
  },
};
```

3. 修改 `main.js` 文件。

```
import vueComposition from "@vue/composition-api"
Vue.use(vueComposition)
```

4. 修改 **TUICallKit/pages/call.vue** 文件。

```
<template>
<view>
  <TUICallKit></TUICallKit>
  <view class="loginBox">
    <input
      class="input-box"
      v-model="userID"
      :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
      placeholder-style="color:#BBBBBB;"
    />
    <view class="login">
      <button
        class="loginBtn"
        @click="!isLogin ? loginHandler() : callHandler()"
      >
        {{ !isLogin ? "登录" : "呼叫" }}
      </button>
    </view>
  </view>
</view>
</template>
<script>
import * as GenerateTestUserSig from "../debug/GenerateTestUserSig-es.js";
import { TUICallKitServer } from "../src/index";
import TUICallKit from "../src/Components/TUICallKit.vue";
export default {
  components: {
    TUICallKit,
  },
  data() {
    return {
      isLogin: false,
      userID: "",
    };
  },
  methods: {
    async loginHandler() {
      if (!this.userID) return;
      const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
        userID: this.userID,
      });
      await TUICallKitServer.init({
        sdkAppID: SDKAppID,
        userID: this.userID,
        userSig: userSig,
      });
      this.isLogin = true;
      this.userID = "";
    },

    async callHandler() {
      await TUICallKitServer.calls({
        userIDList: [this.userID],
        type: 2,
```

```
        });
      },
    },
  };
  </script>
  <style>
  .loginBox {
    margin-top: 200px;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
  }

  input {
    display: flex;
    font-size: 20px;
  }

  .login {
    width: 100vw;
    bottom: 5vh;
    margin: 70rpx;
  }

  .login button {
    width: 80%;
    background-color: #006eff;
    border-radius: 50px;
    color: white;
  }
  </style>
```

5. 修改 pages/index/index.vue 文件。

> **说明：**
> 此处的代码的作用是，主包路由跳转到 TUICallKt 分包页面，注意需要根据您的业务进行调整，此处以空项目举例。

```
<template>
  <view>
    <button @click="jumpUrl">跳转分包</button>
  </view>
</template>

<script>
export default {
  methods: {
    jumpUrl() {
      uni.navigateTo({ url: "/TUICallKit/pages/call" });
    },
  },
};
```

```
</script>
```

警告：

- 如果使用 HBuilder x 4.x 以上版本运行 Vue3 出现如下报错：

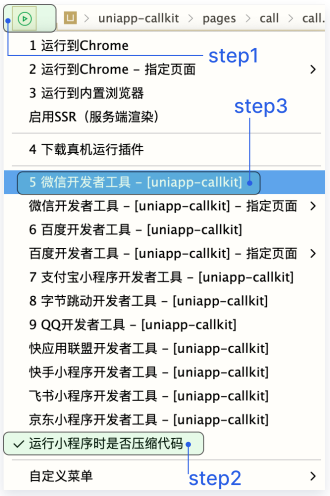
```
11:07:43.690 [plugin:uni:mp-inject] Unexpected token `@`. Expected identifier, string literal, numeric literal or [ for the computed key
11:07:43.700 at TUICallKit/src/TUICallService/serve/callManager.ts:1:0
```

- 可在 tsconfig.json 中添加相关配置，示例：

```
{
  "compilerOptions": {
    "experimentalDecorators": true
  }
}
```

步骤七：运行到微信开发者工具

1. 运行到微信开发者工具。



2. 请在本地设置里面勾选上“不校验合法域名、web-view (业务域名)、 TLS 版本以及 HTTPS 证书”。

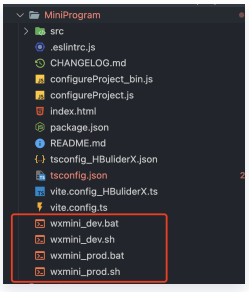
警告：

如果不勾选该条目，则会在控制台出现如下错误。

3. 单击**清缓存 > 全部清除**，避免开发者工具的缓存造成渲染异常。

4. 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）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在您的**项目根目录**：wxmini_uniapp_script 。



Windows

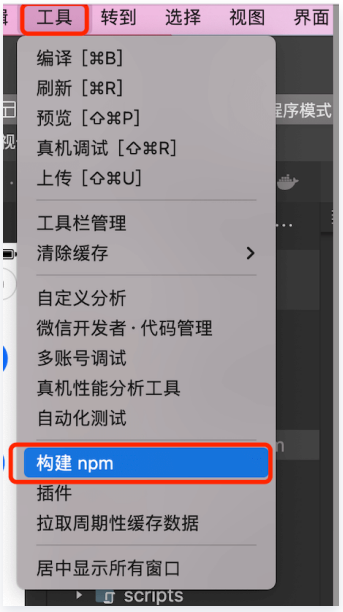
```
# 此处使用的路径为默认路径，如果您对此路径进行了修改，需要手动对脚本路径进行调整
./wxmini_dev.bat
```

Mac

```
# 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。

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



6. 编译小程序。

7. 该项目快速集成后的预期效果图。

步骤八：拨打您的第一通电话

1. 请单击**预览**，扫描二维码，在真机环境使用小程序。

2. 登录后，请输入呼叫用户 ID，拨打您的第一通电话。具体效果如下图所示：

> **注意：**
> 第一次使用小程序通话，需要获取摄像头和麦克风权限。

更多特性

- 设置昵称、头像
- 自定义铃声
- 群组通话
- 悬浮窗

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 。

技术咨询

了解更多详情您可前往 腾讯云通信官方社群 进行咨询和反馈。

uni-app（客户端）

最近更新时间：2025-05-16 17:17:42

本文将介绍如何快速完成 TUICallKit 组件的接入，您将在10分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

TUICallKit Demo 体验

- TUICallKit 插件地址：TUICallKit 插件链接。
- 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

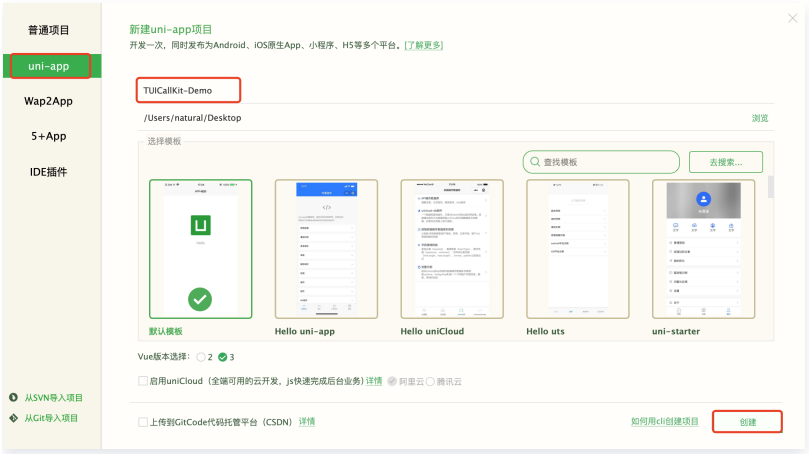
- HbuilderX 版本要求：HbuilderX 版本 ≥ 3.94。
- 插件调试说明：原生插件暂不支持模拟器调试。
- iOS 设备要求：iOS 系统 ≥ 9.0，支持音视频通话的真机设备。
- Android 设备要求：Android 系统 ≥ 5.0（SDK API Level 21），支持音视频的真机设备，允许 USB 调试。

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

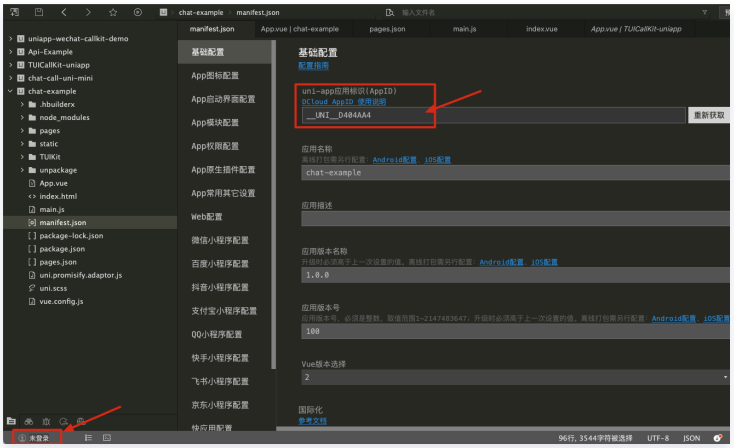
打开 Hbuilderx 开发工具，点击新建 uni-app 项目：项目名称 (TUICallKit-Demo)。



步骤三：下载并导入 TUICallKit 插件

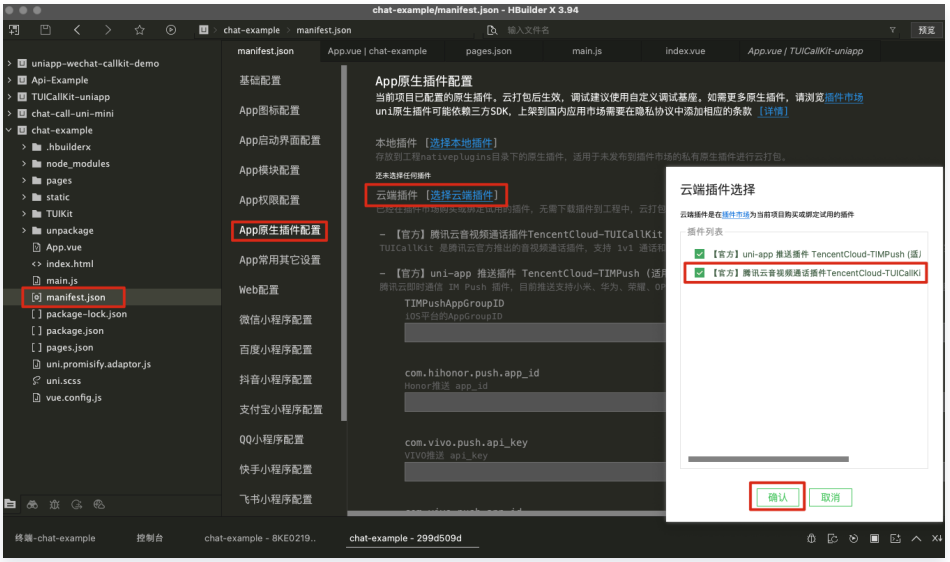
1. 创建项目，生成 uin-app 应用标识（AppID）

   打开 HbuilderX，点击左下角登录 uni-app 账号（无账号请先注册）。完成登录后，点击项目的 manifest.json 文件，生成生成 uni-app 应用标识（AppID）。



2. 访问 TencentCloud-TUICallKit 插件，在插件详情页中购买插件（**免费**），购买插件时选择对应的 AppID，绑定正确的包名。

3. 在 `TUICallKit-Demo 项目` 中导入插件。



步骤四：使用 TUICallKit 插件

1. 在 `TUICallKit-Demo/pages/index/index.vue` 中引入下面代码，该段代码功能：登录 TUICallKit 组件、拨打 1v1 视频通话。

```
<template>
    <view class="container">
        <input type="text" v-model="inputID" :placeholder=" isLogin ? 'please enter a caller
userID' : 'please enter your login userID' " />
        <text v-show="isLogin"> your userID: {{ userID }} </text>
        <button v-show="!isLogin" @click="handleLogin"> Login </button>
        <button v-show="isLogin" @click="handleCall"> start call </button>
    </view>
</template>
<script>
    const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit'); //【1】import
TUICallKit plugin
    uni.$TUICallKit = TUICallKit;
    import { genTestUserSig } from '../../debug/GenerateTestUserSig.js'
    export default {
        data() {
            return {
                inputID: '',
                isLogin: false,
                userID: '',
            }
        },
        methods: {
            handleLogin() {
                this.userID = this.inputID;
                const { userSig, sdkAppID: SDKAppID } = genTestUserSig(this.userID);
                const loginParams = { SDKAppID, userID: this.userID, userSig };   // apply
SDKAppID、userSig
                //【2】Login
                uni.$TUICallKit.login( loginParams, res => {
                        if (res.code === 0) {
                            this.isLogin = true;
                            this.inputID = '';
                            console.log('[TUICallKit] login success.');
                        } else {
                            console.error('[TUICallKit] login failed, failed message = ',
res.msg, params);
                        }
                    }
                );
            },
            handleCall() {
                try {
                    const callParams = {
                        userIDList: [this.inputID],
                        callMediaType: 2,   // 1 -- audio call, 2 -- video call
                        callParams: { roomID: 234, strRoomID: '2323423', timeout:30 },
                    };
                    //【3】start 1v1 video call
                    uni.$TUICallKit.calls( callParams, res => {
                            console.log('[TUICallKit] call params: ', JSON.stringify(res));
                        }
                    );
                    this.inputID = '';
```

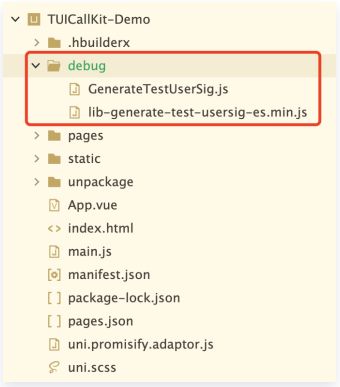
```
                } catch (error) {
                    console.log('[TUICallKit] call error: ', error);
                }
            }
        }
    }
</script>
<style>
.container {
    margin: 30px;
}
.container input {
    height: 50px;
    border: 1px solid;
}
.container button {
    margin-top: 30px;
}
</style>
```

2. 填写 SDKAppID、SecretKey、userSig 参数。

客户端生成 userSig

由于 UserSig 有时效性，如果您需要长期使用 TUICallKit，推荐您采用该方法。

1. **单击下载 debug 文件夹**，将 debug 目录复制到您的项目，如下图所示：



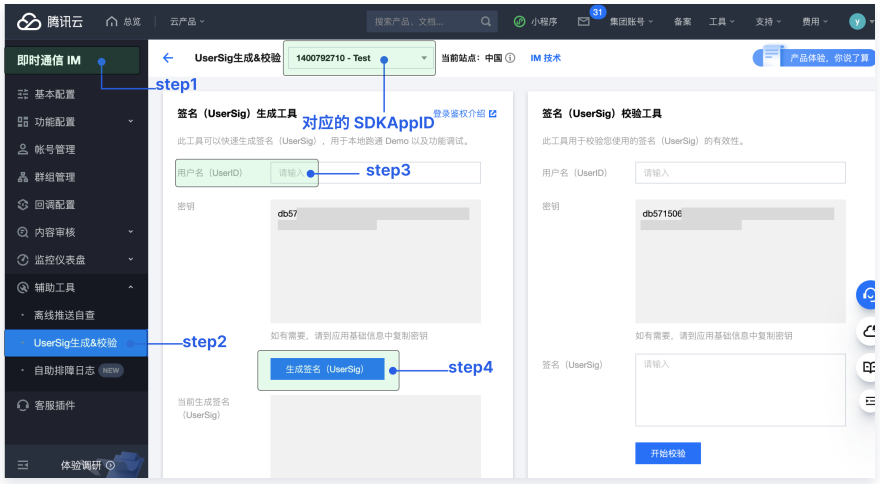
2. 填写 `TUICallKit-Demo/debug/GenerateTestUserSig.js` 文件的 `SDKAppID` 、 `SecretKey` （参见 **开通服务** ）

GenerateTestUserSig.js

```
import LibGenerateTestUserSig from './lib-generate-test-usersig-es.min.js';
/** ... */

const SDKAPPID = 0;
/** ... */

const EXPIRETIME = 604800;
/** ... */

const SECRETKEY = '';
/* ... */

export function genTestUserSig(userID) {
  const generator = new LibGenerateTestUserSig(SDKAPPID, SECRETKEY, EXPIRETIME);
  const userSig = generator.genTestUserSig(userID);

  return {
    sdkAppID: SDKAPPID,
    userSig,
  };
}
```

控制台生成 userSig

如果您想要快速体验 TUICallKit，您可以通过控制台中的 **辅助工具** 生成一个临时可用的 UserSig。



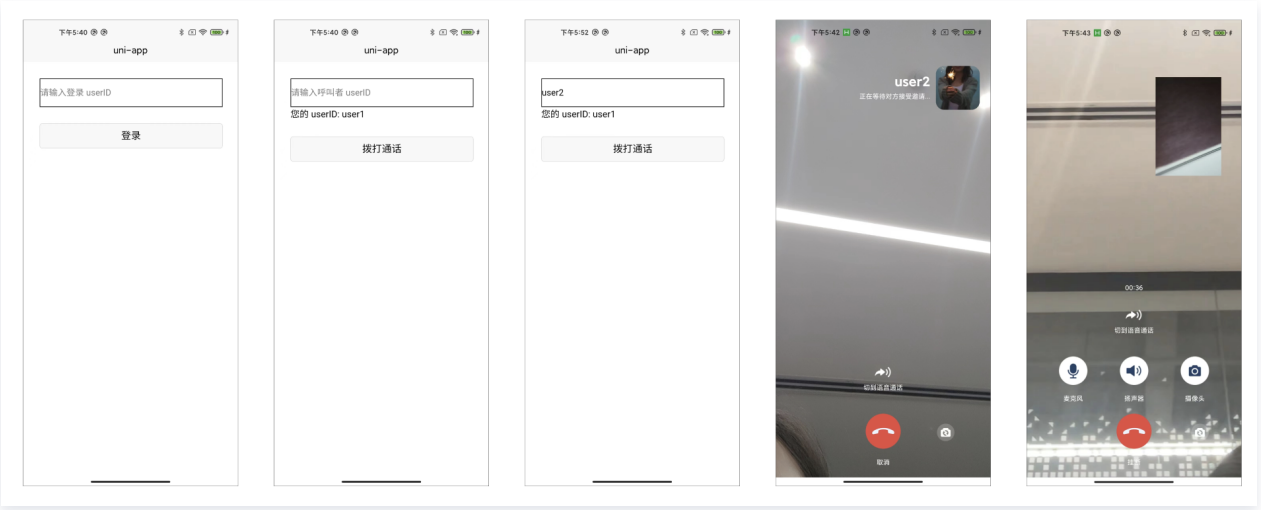
如果您采用的是**控制台生成**，那么需要在 `TUICallKit-Demo/pages/index/index.vue` **文件中，赋值这里的** `SDKAppID、userSig`。

步骤五：拨打您的第一通电话

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. 拨打 1v1 视频通话具体效果如图所示。



更多特性

- 设置昵称、头像
- 群组通话
- 悬浮窗
- 自定义铃声
- 监听通话状态

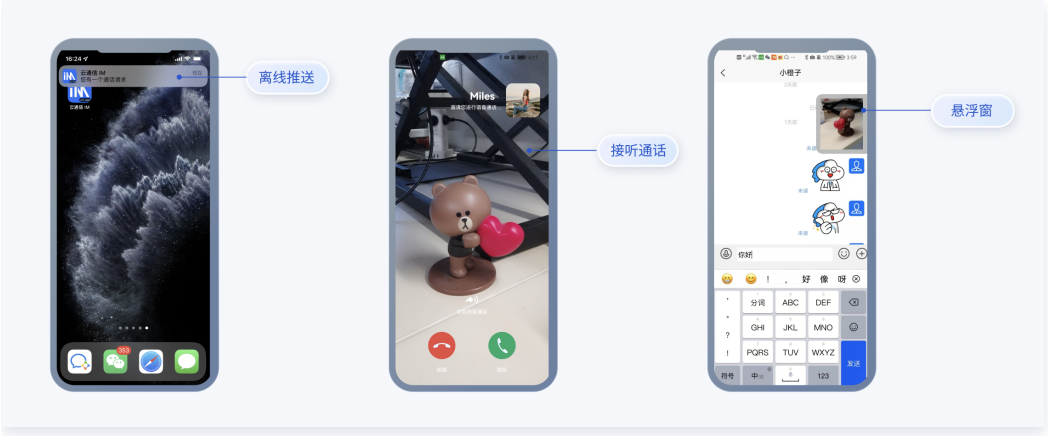
常见问题

如果您的接入和使用中遇到问题，请参见 常见问题 。

相关案例-在线客服场景

我们提供了**在线客服场景**的相关源码，建议您 下载在线客服场景 并集成体验。该场景提供了示例客服群 + 示例好友的基础模板，实现功能包括：

- 支持发送文本消息、图片消息、语音消息、视频消息等常见消息。
- 支持双人语音、视频通话功能。
- 支持创建群聊会话、群成员管理等。



技术咨询

了解更多详情您可加入 腾讯云通信官方社群 进行咨询和反馈。

小程序插件

最近更新时间：2025-07-28 15:37:12

> **注意：**
> 请您花费一到两分钟仔细阅读一下源码集成和插件集成的对比。

集成方式 / 场景描述	插件集成	源码集成
是否占用主包体积	不占用	占用
接入成本	简单	耗时
TUICallKIt 迭代	插件如果有更新，会及时推送给插件使用者 小程序插件快速更新功能说明。	无法推送给使用者，需要用户持续关注 发布日志。
升级成本	由于不涉及源码，没有任何升级成本。	如果您对源码有修改，升级成本非常巨大。
是否适用于 Vue/React 的框架环境	• uniapp 框架使用小程序插件参见 使用小程序插件 文档 • taro 框架使用小程序插件参见 使用小程序原生第三方组件和插件文档	不适用

本文将介绍如何快速完成 TUICallKit 组件的接入，您将在三分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

小程序 Demo 体验

- 如果您想要直接体验音视频通话小程序，单击 Demo 体验，扫描小程序二维码。
- 如果您想要直接跑通一个新工程，请直接阅读 微信小程序 Demo 快速跑通 。

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：2.10.0。

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**上运行体验。

插件说明

本文涉及的小程序插件是一个 社交 > 直播 类目的小程序插件，**仅限国内注册主体的电商平台、有资质的教育类目小程序使用**。详细的类目要求如下：

一级类目	二级类目
电商平台	电商平台
	网上竞价平台（非文物）
	网上竞价平台（文物）
教育	学历教育（培训机构）
	学历教育（学校）
	驾校培训
	驾校平台
	教育平台
	在线视频课程

注意：
1. 微信小程序的主体必须为非个人主体类型，否则无法使用 TUICallKit 插件。以上表格**仅提供参考**，详细的微信小程序类目及申请资质要求需以微信最新的 微信非个人主体小程序开放的服务类目 为准。
2. 微信小程序的类目即为微信小程序的服务场景，在小程序后台的**设置 > 基本设置 > 服务类目**中，可以选择符合小程序功能的类目。所选类目需符合小程序的实际应用场景，否则在提交审核后会被驳回申请。

小程序开发准备

步骤一：开通企业类小程序

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：

步骤二：在小程序控制台开启实时音视频接口

- 小程序推拉流标签使用权限暂时只开放给有限类目，具体支持类目参见该地址 。
- 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。

步骤三：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。

- 将以下域名添加到 socket 合法域名：

域名	说明	是否必须

`wss://${SDKAppID}w4c.my-imcloud.com`	v3.4.6起，SDK 支持独立域名，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`	必须
`wss://wss.im.qcloud.com`	Web IM 业务域名	必须
`wss://wss.tim.qq.com`	Web IM 业务域名	必须
`wss://wssv6.im.qcloud.com`	Web IM 业务域名	必须

- 将以下域名添加到 **request 合法域名**：

域名	说明	是否必须
`https://web.sdk.qcloud.com`	Web IM 业务域名	必须
`https://boce-cdn.my-imcloud.com`	Web IM 业务域名	必须
`https://api.im.qcloud.com`	Web IM 业务域名	必须
`https://events.im.qcloud.com`	Web IM 业务域名	必须
`https://webim.tim.qq.com`	Web IM 业务域名	必须
`https://wss.im.qcloud.com`	Web IM 业务域名	必须
`https://wss.tim.qq.com`	Web IM 业务域名	必须

- 将以下域名添加到 **uploadFile 合法域名**：

域名	说明	是否必须
`https://${SDKAppID}-cn.rich.my-imcloud.com`	从 2024年9月10日起，新增应用默认分配 cos 独立域名。例如您的 SDKAppID 是 1400xxxxxx，则 cos 独立域名为：`https://1400xxxxxx-cn.rich.my-imcloud.com`	必须
`https://cn.rich.my-imcloud.com`	文件上传域名	必须
`https://cn.imrich.qcloud.com`	文件上传域名	必须
`https://cos.ap-shanghai.myqcloud.com`	文件上传域名	必须
`https://cos.ap-shanghai.tencentcos.cn`	文件上传域名	必须
`https://cos.ap-guangzhou.myqcloud.com`	文件上传域名	必须

- 将以下域名添加到 **downloadFile 合法域名**：

域名	说明	是否必须
`https://${SDKAppID}-cn.rich.my-imcloud.com`	从 2024年9月10日起，新增应用默认分配 cos 独立域名。	必须

	例如您的 SDKAppID 是 1400xxxxxx，则 cos 独立域名为：`https://1400xxxxxx-cn.rich.my-imcloud.com`	
`https://cn.rich.my-imcloud.com`	文件下载域名	必须
`https://cn.imrich.qcloud.com`	文件下载域名	必须
`https://cos.ap-shanghai.myqcloud.com`	文件下载域名	必须
`https://cos.ap-shanghai.tencentcos.cn`	文件下载域名	必须
`https://cos.ap-guangzhou.myqcloud.com`	文件下载域名	必须

- 如下图所示，服务器域名配置：

步骤四：添加插件

若是首次使用小程序插件，请阅读了解微信官方文档小程序 使用插件 内容。在使用插件前，首先要在小程序管理后台的**设置 > 第三方设置 > 插件管理**中**添加插件**。开发者可登录小程序管理后台，搜索 TRTC音视频通话 申请添加插件。单击**查看详情**，可查看插件的“基本信息”、“相关讨论”等。其中“基本信息”页包括小程序插件 AppID、更新记录等。

> **注意：**
> 如果无法搜索到该插件，可以尝试直接点击 此处添加 。

TUICallKit 插件集成

步骤一：配置分包

1. 修改 app.json 文件，注册分包，添加插件并移除 skyline 相关配置。

```
"subPackages": [
  {
    "root": "TUICallKit",
    "name": "TUICallKit",
    "pages": [
      "pages/call/call"
    ],
    "plugins": {
      "TUICallKit-plugin": {
        "version": "latest",
        "provider": "wx6546805a14bb7ef9"
      }
    },
    "independent": false
  }
]
```

警告：

微信开发者工具新的渲染引擎 skyline 和 live-pusher 存在兼容性问题，需要将以下代码从 app.json 中去除。

```
"renderer": "skyline",
"rendererOptions": {
  "skyline": {
    "defaultDisplayBlock": true,
    "disableABTest": true,
    "sdkVersionBegin": "3.0.0",
    "sdkVersionEnd": "15.255.255"
  }
```

```
    },
    "componentFramework": "glass-easel",
    "sitemapLocation": "sitemap.json",
    "lazyCodeLoading": "requiredComponents"
```

2. 修改分包 **TUICallKit/pages/call/call** 页面 使用插件。

wxml 文件

```
<view class="container">
  <TUICallKit></TUICallKit>
  <view class="box">
    <view class="input-box">
      <input type="text" maxlength="20" placeholder="{{isLogin?'请输入呼叫者userID':'请输入登录者userID' }}" value="{{userID}}" bindinput='bindInputUserID' placeholder-style="color:#BBBBBB;" />
    </view>
    <view class='login'>
      <button class='loginBtn' bindtap="{{isLogin?'call':'login'}}">{{isLogin?'呼叫':'登录'}}</button>
    </view>
  </view>
</view>
```

json 文件

```
{
  "usingComponents": {
    "TUICallKit": "plugin://TUICallKit-plugin/TUICallKit-plugin"
  }
}
```

js 文件

```
// setup 1: 导入插件
const plugin = requirePlugin('TUICallKit-plugin')
Page({
  data: {
    userID: "",
    isLogin: false,
  },

  bindInputUserID(e) {
    this.setData({
```

```
      userID: e.detail.value,
    });
  },

  // setup 2: 调用登录方法
  async login() {
    const userID = this.data.userID;
    if (!userID) return;
    const UserSigConfig = {
      userID: userID,
      SDKAppID: 0, // 填入您的 SDKAppID
      SecretKey: '' // 填入您的 SecretKey
    }
    const {
      userSig
    } = plugin.genTestUserSig(UserSigConfig);
    plugin.getTUICallKitServer().init({
      sdkAppID: UserSigConfig.SDKAppID,
      userID: UserSigConfig.userID,
      userSig: userSig
    })
    wx.showToast({
      title: "登录成功",
      icon: "error",
    });
    this.setData({
      isLogin: true,
      userID: "",
    });
  },

  // setup 3: 调用呼叫方法
  async call() {
    await plugin.getTUICallKitServer().call({
      userID: this.data.userID,
      type: 2,
    });
  },
});
```

wxss 文件

```
.container {
  width: 100vw;
  height: 100vh;
}


.box {
  flex: 1;
  width: 100vw;
```

```
  margin-top: -40px;
  background: #ffffff;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}


input {
  display: flex;
  font-size: 20px;
  width: 60vw;
}


.login {
  display: flex;
  width: 100vw;
  text-align: center;
  bottom: 5vh;
  margin: 70rpx;
}


.login button {
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
```

3. 修改 **pages/index/index** 文件。

> ⚠ **注意：**
> 此处的代码的作用是，主包路由跳转到 TUICallKt 分包页面，注意需要根据您的业务进行调整，此处以空项目举例。

wxml 文件

```
<view class="container">
  <button bind:tap="jumpUrl">跳转分包</button>
</view>
```

js 文件

```
Page({
```

```
  data: {
  },
  jumpUrl() {
    wx.navigateTo({ url: "/TUICallKit/pages/call/call" });
  },
})
```

wxss 文件

```
.container {
  margin-top: 100px;
}
```

步骤二：拨打您的第一通电话

- 请单击预览，扫描二维码，在真机环境使用小程序。

- 登录后，请输入呼叫用户 ID，拨打您的第一通电话。具体效果如下图所示：

> **注意：**
> 第一次使用小程序通话，需要获取摄像头和麦克风权限。

更多特性

- 设置昵称、头像
- 自定义铃声
- 群组通话

常见问题

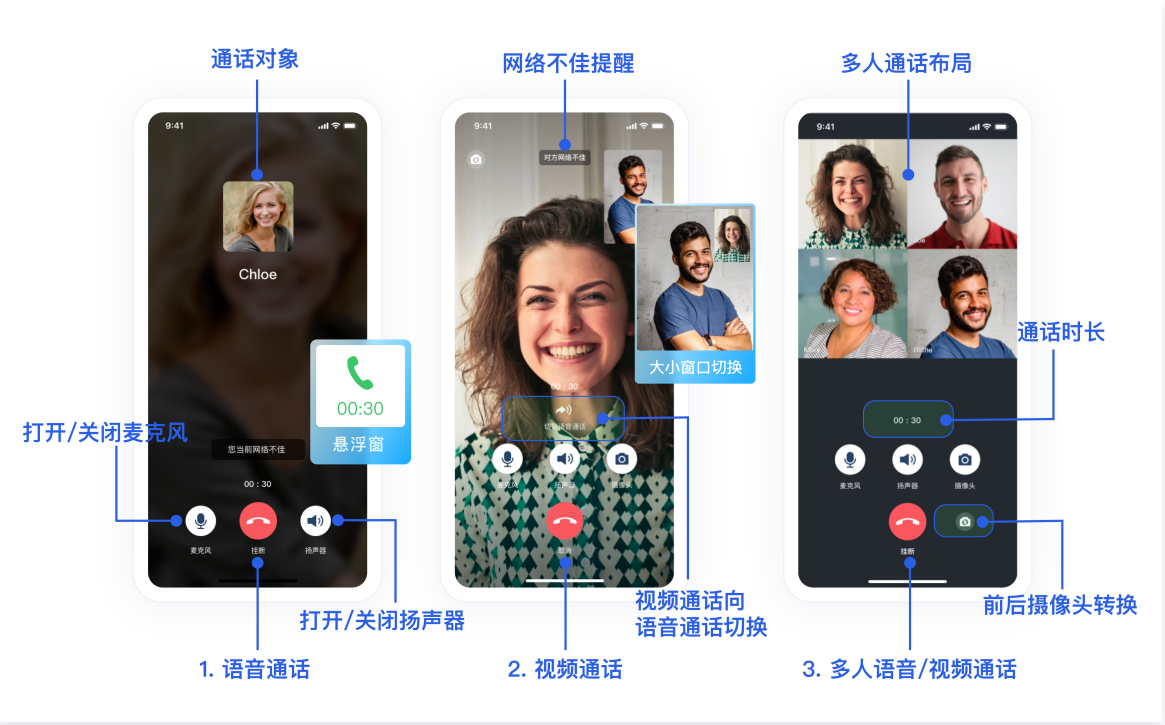
> **警告：**
> **小程序插件无法针对源码进行修改，无法使用全局监听功能，也无法搭配聊天组件 TUIKit 使用**，如果您有上述需求，请参见 小程序源码集成 。

如果您的接入和使用中遇到问题，请参见 常见问题 。

微信小程序

最近更新时间：2025-03-28 09:50:22

本文将介绍如何快速完成 TUICallKit 组件的接入，跟随本文档，您将在半小时内得到一个包含完备 UI 界面的视频通话小程序。基本功能如下图所示：



小程序 Demo 体验

- 如果您想要直接体验音视频通话小程序，单击 Demo 体验，扫描小程序二维码。
- 如果您想要直接跑通一个新工程，请直接阅读 微信小程序 Demo 快速跑通 。

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：2.10.0。

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**上运行体验。

小程序开发准备

步骤一：开通企业类小程序

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：

步骤二：在小程序控制台开启实时音视频接口

- 小程序推拉流标签使用权限暂时只开放给有限类目，具体支持类目参见该地址 。
- 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。

步骤三：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。

- 将以下域名添加到 socket 合法域名：

域名	说明	是否必须

wss://${SDKAppID}w4c.my-imcloud.com	v3.4.6起，SDK 支持独立域名，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	必须
wss://wss.im.qcloud.com	Web IM 业务域名	必须
wss://wss.tim.qq.com	Web IM 业务域名	必须
wss://wssv6.im.qcloud.com	Web IM 业务域名	必须

- 将以下域名添加到 **request 合法域名**：

域名	说明	是否必须
https://web.sdk.qcloud.com	Web IM 业务域名	必须
https://boce-cdn.my-imcloud.com	Web IM 业务域名	必须
https://api.im.qcloud.com	Web IM 业务域名	必须
https://events.im.qcloud.com	Web IM 业务域名	必须
https://webim.tim.qq.com	Web IM 业务域名	必须
https://wss.im.qcloud.com	Web IM 业务域名	必须
https://wss.tim.qq.com	Web IM 业务域名	必须

- 将以下域名添加到 **uploadFile 合法域名**：

域名	说明	是否必须
https://${SDKAppID}-cn.rich.my-imcloud.com	从 2024年9月10日起，新增应用默认分配 cos 独立域名。例如您的 SDKAppID 是 1400xxxxxx，则 cos 独立域名为：https://1400xxxxxx-cn.rich.my-imcloud.com	必须
https://cn.rich.my-imcloud.com	文件上传域名	必须
https://cn.imrich.qcloud.com	文件上传域名	必须
https://cos.ap-shanghai.myqcloud.com	文件上传域名	必须
https://cos.ap-shanghai.tencentcos.cn	文件上传域名	必须
https://cos.ap-guangzhou.myqcloud.com	文件上传域名	必须

- 将以下域名添加到 **downloadFile 合法域名**：

域名	说明	是否必须
https://${SDKAppID}-cn.rich.my-imcloud.com	从 2024年9月10日起，新增应用默认分配 cos 独立域名。	必须

	例如您的 SDKAppID 是 1400xxxxxx，则 cos 独立域名为：`https://1400xxxxxx-cn.rich.my-imcloud.com`	
`https://cn.rich.my-imcloud.com`	文件下载域名	必须
`https://cn.imrich.qcloud.com`	文件下载域名	必须
`https://cos.ap-shanghai.myqcloud.com`	文件下载域名	必须
`https://cos.ap-shanghai.tencentcos.cn`	文件下载域名	必须
`https://cos.ap-guangzhou.myqcloud.com`	文件下载域名	必须

- 如下图所示，服务器域名配置：

TUICallKit 源码集成

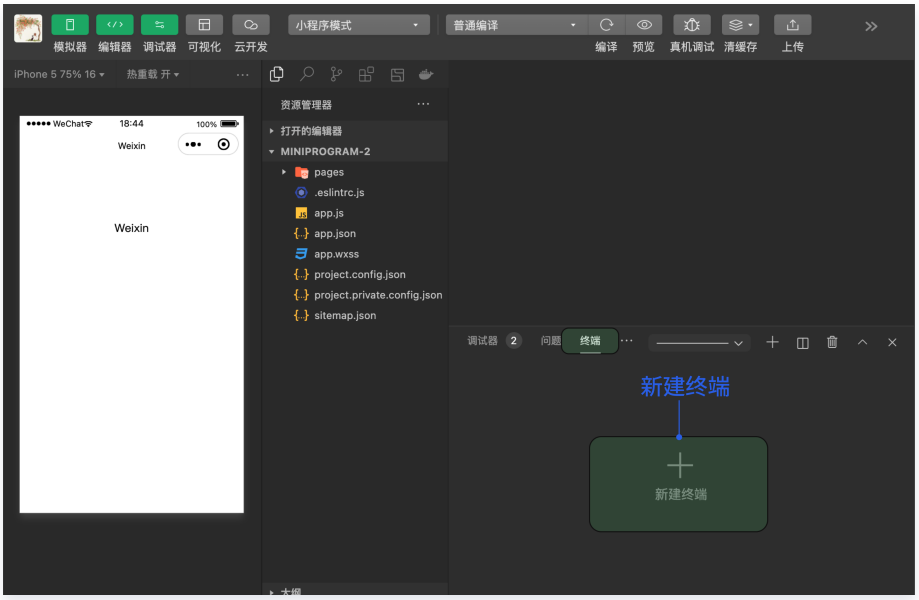
步骤一：开通服务

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：创建小程序项目

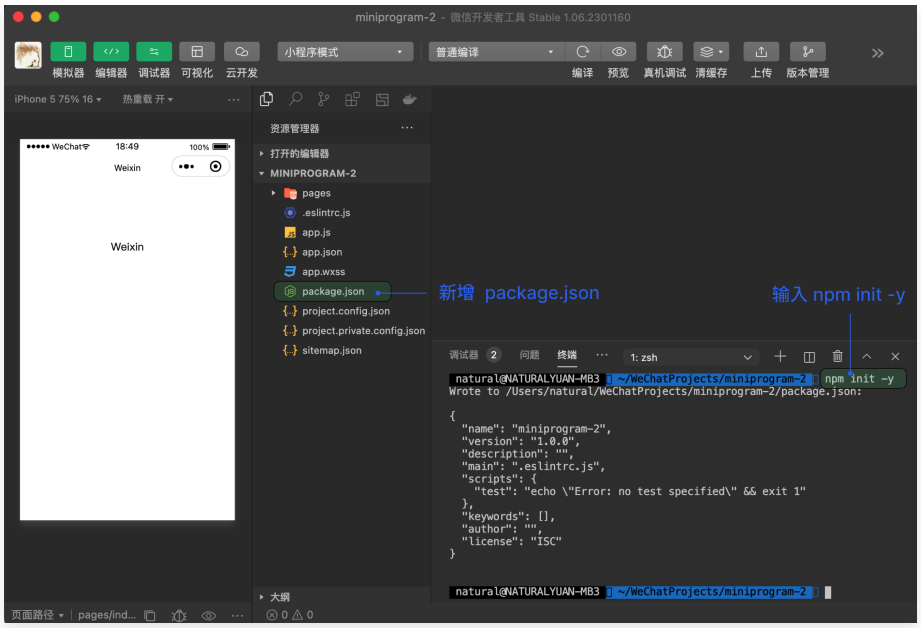
1. 在微信开发者工具上创建一个小程序项目，选择**不使用模板**。

2. 新建终端。



3. 执行 `npm init -y` 命令生成 `package.json` 文件。

```
npm init -y
```



步骤三：下载并导入 TUICallKit 组件

1. 下载 TUICallKit 组件。

```
npm i @tencentcloud/call-uikit-wx
```

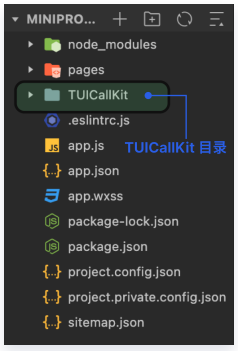
MacOS 端

```
mkdir -p ./TUICallKit && cp -r node_modules/@tencentcloud/call-uikit-wx/ ./TUICallKit && cp
node_modules/@tencentcloud/call-engine-wx/RTCCallEngine.wasm.br ./static
```

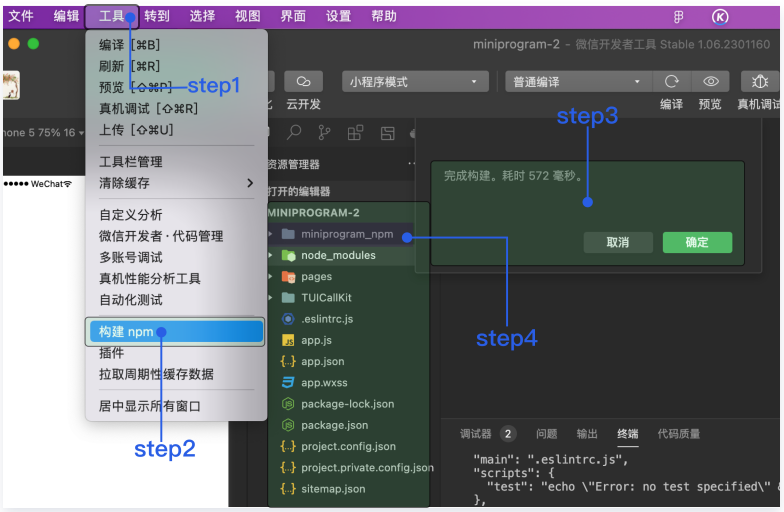
Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-wx\ .\TUICallKit /i /e
xcopy node_modules\@tencentcloud\call-engine-wx\RTCCallEngine.wasm.br .\static
```

2. 执行完以上命令后，您的目录下生成 TUICallKit 文件夹，其中包含有 TUICallKit 组件。目录如下：

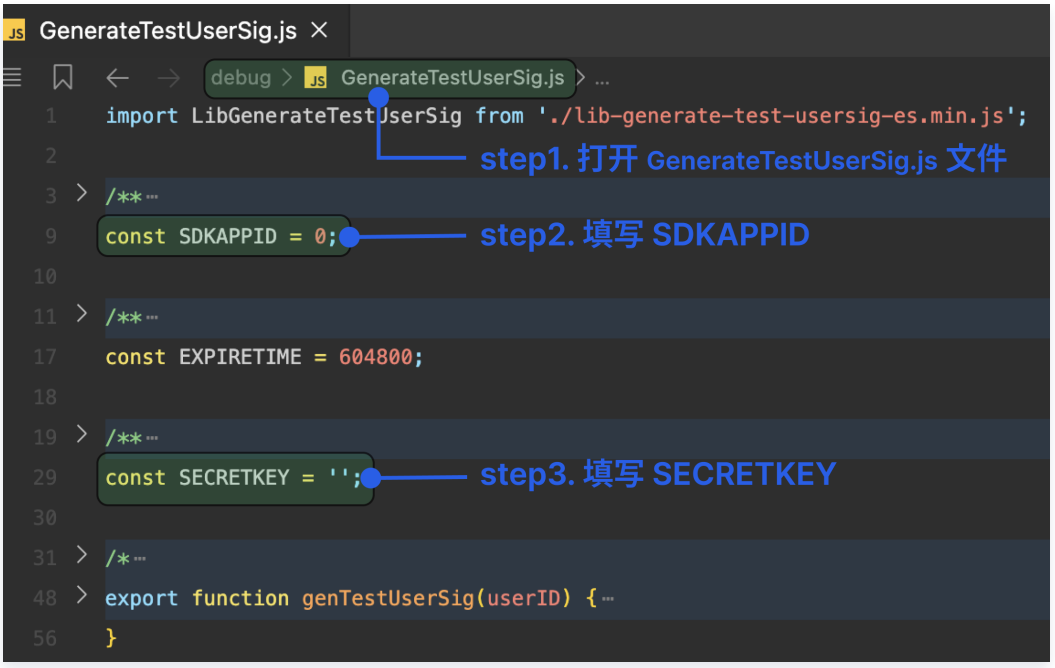


3. 构建 npm，打开微信开发者工具单击工具 > 构建 npm，新增 miniprogram_npm 目录。目录如下：



步骤四：填写 SDKAPPID 以及 SECRETKEY

修改 TUICallKit/debug/`GenerateTestUserSig-es.js` 文件 的 SDKAPPID 以及 SECRETKEY。



步骤五：调用 TUICallKit 组件

1. 修改 `app.json` 文件，添加如下代码，新增全局监听页面。

```
{
  "pages": [
    "pages/index/index",
    "TUICallKit/pages/globalCall/globalCall"
  ],
  "window": {
    "navigationBarTextStyle": "black",
    "navigationStyle": "custom"
  },
  "style": "v2"
}
```

2. 修改 `index` 文件夹下的文件。

index.wxml

```
<view class="container">
  <view class="box">
    <view class="input-box">
      <input type="text" maxlength="20" placeholder="{{isLogin?'请输入呼叫者userID':'请输入登录者
userID' }}" value="{{userID}}" bindinput='bindInputUserID' placeholder-style="color:#BBBBBB;"
/>
    </view>
    <view class='login'>
      <button class='loginBtn' bindtap="{{isLogin?'call':'login'}}">{{isLogin?'呼叫':'登录'}}
</button>
    </view>
  </view>
</view>
```

index.js

```
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/TUICallService/index";
// 导入 CallManager 模块，使您的应用具有全局监听来电的能力
import { CallManager } from "../../TUICallKit/TUICallService/serve/callManager";
import * as GenerateTestUserSig from "../../TUICallKit/debug/GenerateTestUserSig-es.js";
wx.CallManager = new CallManager();
Page({
  data: {
    userID: "",
    isLogin: false,
  },

  bindInputUserID(e) {
    this.setData({
      userID: e.detail.value,
    });
  },

  async login() {
    const userID = this.data.userID;
    if (!userID) return;
    const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
      userID: userID,
    });
    await wx.CallManager.init({
      sdkAppID: SDKAppID,
      userID: userID,
      userSig: userSig,
      globalCallPagePath: "TUICallKit/pages/globalCall/globalCall",
    });
    wx.showToast({
      title: "登录成功",
      icon: "error",
```

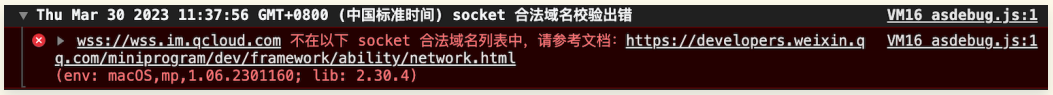
```
    });
    this.setData({
      isLogin: true,
      userID: "",
    });
  },

  async call() {
    await TUICallKitServer.calls({
      userIDList: [this.data.userID],
      type: 2,
    });
  },
});
```

index.wxss

```
.container {
  width: 100vw;
  height: 100vh;
}

.box{
  flex: 1;
  width: 100vw;
  margin-top: -40px;
  background: #ffffff;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

input{
  display: flex;
  font-size: 20px;
  width: 60vw;
}

.login {
  display: flex;
  width: 100vw;
  text-align: center;
  bottom: 5vh;
  margin: 70rpx;
}
.login button{
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
```

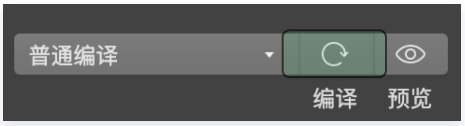
步骤六：编译运行

1. 请在本地设置里面勾选上“不校验合法域名、web-view (业务域名)、 TLS 版本以及 HTTPS 证书”。

> **警告：**
> **如果不勾选该条目，则会在控制台出现如下错误。**
>
> 
>

2. 单击**清缓存 > 全部清除**，避免开发者工具的缓存造成渲染异常。

3. 编译小程序。



4. 该项目快速集成后的预期效果图。

步骤七：拨打您的第一通电话

- 请单击**预览**，扫描二维码，在真机环境使用小程序。

- 登录后，请输入呼叫用户 ID，拨打您的第一通电话。具体效果如下图所示：

> **注意：**
> 第一次使用小程序通话，需要获取摄像头和麦克风权限。

更多特性

- 设置昵称、头像
- 自定义铃声
- 群组通话

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 。

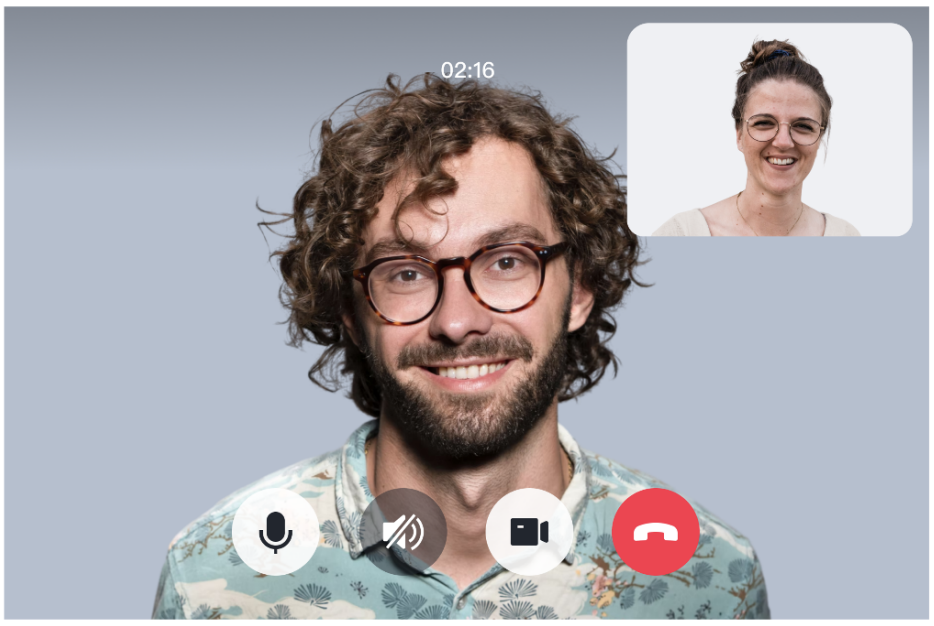
技术咨询

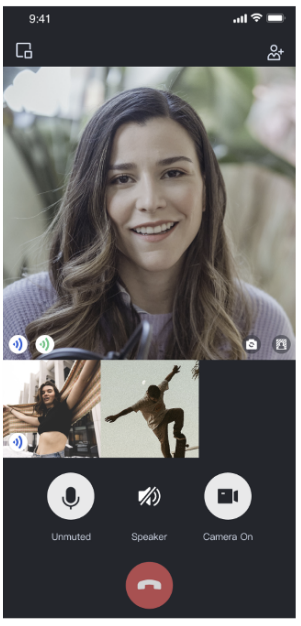
了解更多详情您可 腾讯云通信官方社群 进行咨询和反馈。

Web&H5 (React)

最近更新时间：2025-06-26 15:00:32

本文将介绍如何快速完成 TUICallKit 组件的接入，您将在10分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。





环境准备

- React version 18+.
- Node.js version 16+.
- Modern browser, supporting WebRTC APIs.

步骤一：开通服务

请参见 开通服务，获取 `SDKAppID`、`SecretKey`，它们将在 初始化 TUICallKit 组件 作为**必填参数**使用。

步骤二：下载 TUICallKit 组件

1. 下载 @tencentcloud/call-uikit-react 组件。

```
npm install @tencentcloud/call-uikit-react
```

2. 将 `debug` 目录复制到您的项目目录 `src/debug`，本地生成 userSig 时需要使用。

MacOS

```
cp -r node_modules/@tencentcloud/call-uikit-react/debug ./src
```

Windows

```
xcopy node_modules\@tencentcloud\call-uikit-react\debug .\src\debug /i /e
```

步骤三：初始化 TUICallKit 组件

您可以选择在 `/src/App.tsx` 文件引入示例代码。

1. 引入 call-uikit 相关 API 对象。

```
import { useState } from 'react';
import { TUICallKit, TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-react";
import * as GenerateTestUserSig from "./debug/GenerateTestUserSig-es"; // Refer to Step 3
```

2. 引入<TUICallKit />，该组件包含通话时的完整 UI 交互。

```
return (
  <>
    <span> caller's ID: </span>
    <input type="text" placeholder='input caller userID' value={callerUserID} onChange=
{(event) => setCallerUserID(event.target.value)} />
    <button onClick={init}> step1. init </button> <br />
    <span> callee's ID: </span>
    <input type="text" placeholder='input callee userID' value={calleeUserID} onChange=
{(event) => setCalleeUserID(event.target.value)} />
    <button onClick={call}> step2. call </button>

    {/* 【1】Import the TUICallKit component: Call interface UI */}
    <TUICallKit />
  </>
);
```

3. 调用 TUICallKitServer.init API 登录组件，需要在代码中**填写** `SDKAppID、SecretKey` **两个参数。**

```
const SDKAppID = 0;         // TODO: Replace with your SDKAppID (Notice: SDKAppID is of type
number)
const SDKSecretKey = '';    // TODO: Replace with your SDKSecretKey

const [callerUserID, setCallerUserID] = useState('');
const [calleeUserID, setCalleeUserID] = useState('');

//【2】Initialize the TUICallKit component
const init = async () => {
  const { userSig } = GenerateTestUserSig.genTestUserSig({
    userID: callerUserID,
    SDKAppID,
    SecretKey: SDKSecretKey,
  });
  await TUICallKitServer.init({
    userID: callerUserID,
    userSig,
    SDKAppID,
  });
  alert('TUICallKit init succeed');
}
```

参数	类型	说明
userID	String	**用户的唯一标识符**，由您定义，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	Number	在 Tencent RTC 控制台 创建的音视频应用的唯一标识。
SDKSecretKey	String	在 Tencent RTC 控制台 创建的音视频应用的 SecretKey。
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。

userSig 说明：

- **开发环境：**如果您正在本地跑通 Demo、开发调试，可以采用 `debug` 文件中的 `genTestUserSig`（参考步骤3.2）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。

步骤四：拨打您的第一通电话

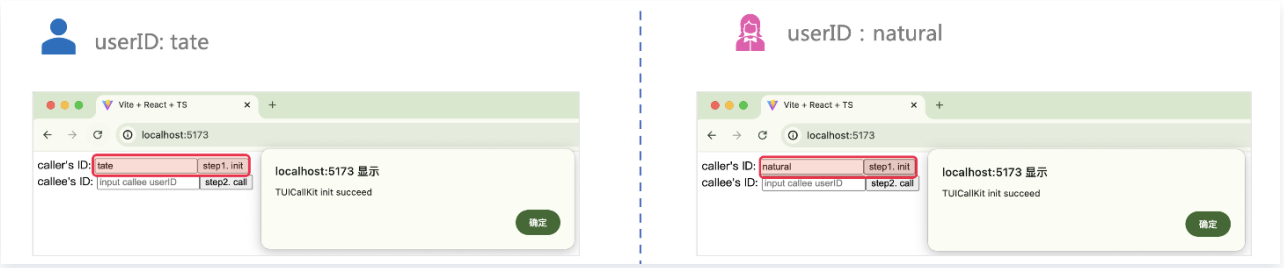
1. 调用 TUICallKitServer.calls API 拨打通话。

```
//【3】Make a 1v1 video call
const call = async () => {
  await TUICallKitServer.calls({
    userIDList: [calleeUserID],
    type: TUICallType.VIDEO_CALL,
  });
};
```

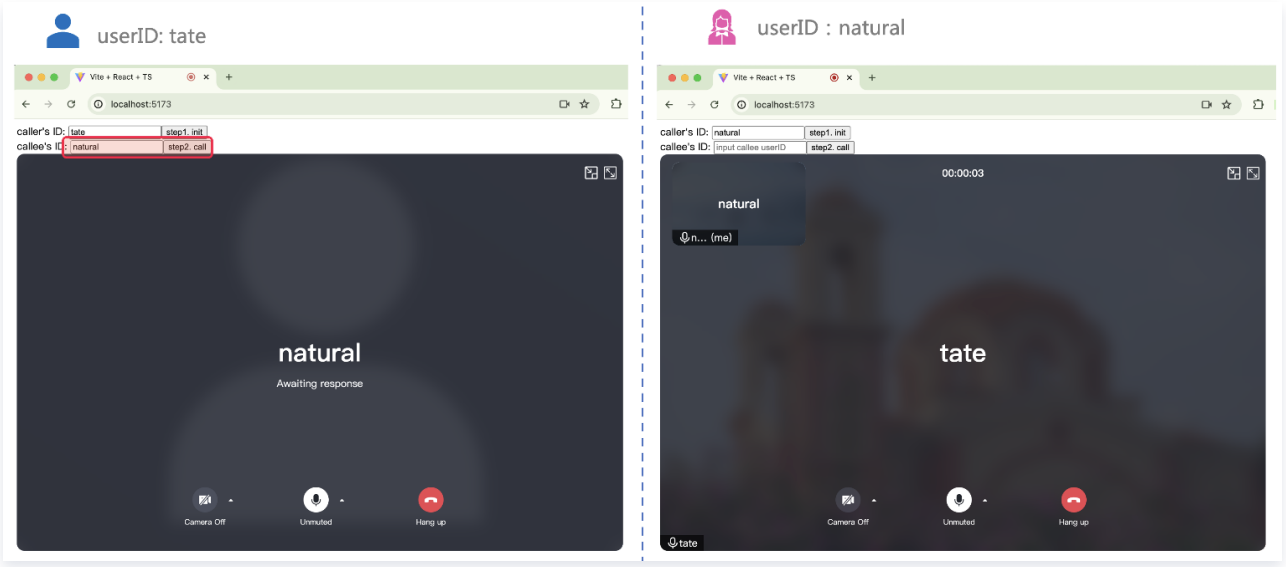
2. 运行项目。

警告：
本地环境请在 localhost 协议下访问，公网体验请在 HTTPS 协议下访问，具体参见 网络访问协议说明 。

3. 打开两个浏览器页面，输入不同的 userID(由您定义) 单击 `step1. init` 登录（主叫方和被叫方）。



4. 两个 userID 都登录成功后，单击 `step2. call` 拨打通话，如果您有通话问题，参见 常见问题 。



更多特性

- 设置昵称、头像
- 群组通话
- 悬浮窗
- 自定义铃声
- 监听通话状态、组件回调事件
- 设置分辨率、填充模式
- 界面定制

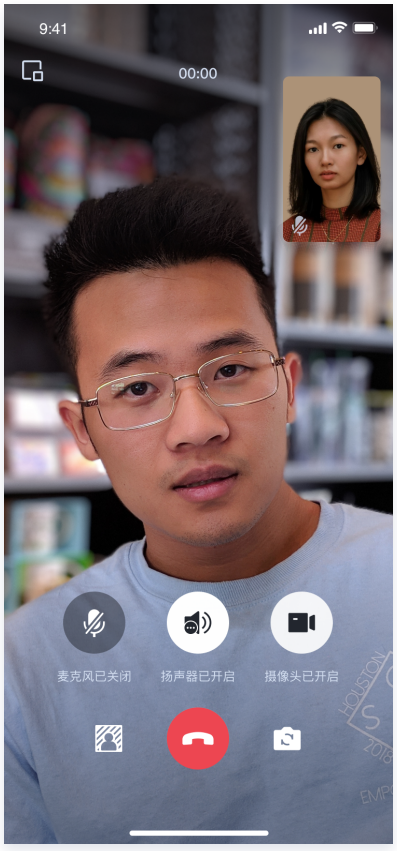
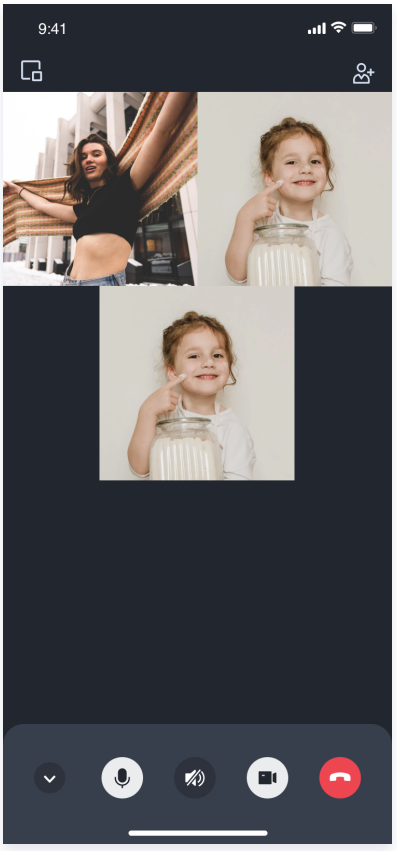
常见问题

- 如果您的接入和使用中遇到问题，请参见 常见问题 。
- 了解更多详情您可 腾讯云通信官方社群 进行咨询和反馈。

Flutter

最近更新时间：2025-06-10 09:37:01

本文将引导您快速地完成 TUICallKit 组件的接入工作。跟随本文档，您可以在10分钟内完成接入，并最终获得一个具备完整用户界面以及音视频通话功能的应用程序。

视频通话	群组通话
	

环境准备

Flutter 3.0 及更高版本。

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，获取 `SDKAppID`、`SDKSecretKey`，它们将在 步骤五 中使用，具体步骤请参见 开通服务。

步骤二：导入 TUICallKit 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_calls_uikit 插件。

```
flutter pub add tencent_calls_uikit
```

步骤三：完成工程配置

Android

1. 如果您需要编译运行在 Android 平台，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。
   - 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 {
            ......
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
rules.pro'
        }
    }
}
```

   - 在工程的 `android/app` 目录下创建 `proguard-rules.pro` 文件，并 `proguard-rules.pro` 文件中添加如下代码：

```
-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 {
      ......
      multiDexEnabled true
    }
}
```

iOS

由于 TUICallKit 会使用 iOS 的音视频功能，您需要授权麦克风和摄像头的使用权限。
授权操作方法：在您的 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>
```

步骤四：设置 navigatorObservers

在 Flutter 应用框架的 navigatorObservers 中添加 TUICallKit.navigatorObserver，以 MateriaApp 框架为例，代码如下：

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

 ......
```

```
class XXX extends StatelessWidget {
  const XXX({super.key});

 @override
  Widget build(BuildContext context) {
    return MaterialApp(
      navigatorObservers: [TUICallKit.navigatorObserver],
      ......
    );
  }
}
```

步骤五：登录 TUICallKit 组件

使用 login 接口完成登录，具体使用可参考如下代码：

```
import 'package:tencent_calls_uikit/tencent_calls_uikit.dart';
import 'package:tencent_calls_uikit/debug/generate_test_user_sig.dart';
......

final String userID    = 'xxxxx';  // 请替换为您的UserId
final int    sdkAppID  = 0;        // 请替换为第一步在控制台得到的SDKAppID
final String secretKey = 'xxxx';   // 请替换为第一步在控制台得到的SecretKey

void login() async {
    String userSig  = GenerateTestUserSig.genTestSig(userID, sdkAppID, secretKey);
    TUIResult result = await TUICallKit.instance.login(sdkAppID, userID, userSig);
    if (result.code.isEmpty) {
      print('Login success');
    } else {
      print('Login failed: ${result.code} ${result.message}');
    }
}
```

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。
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。

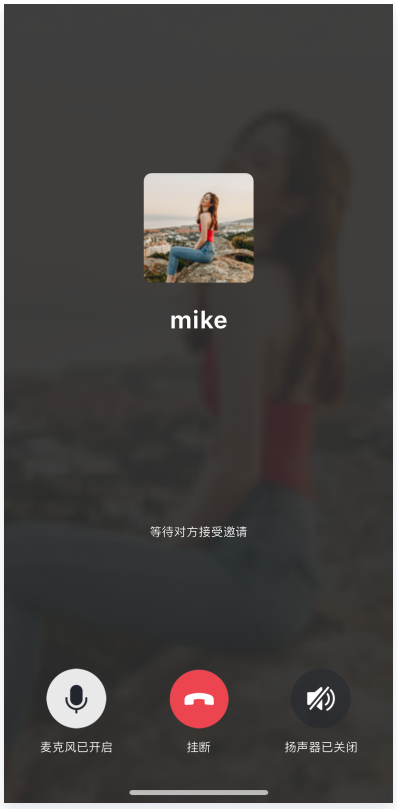
注意：

- **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。

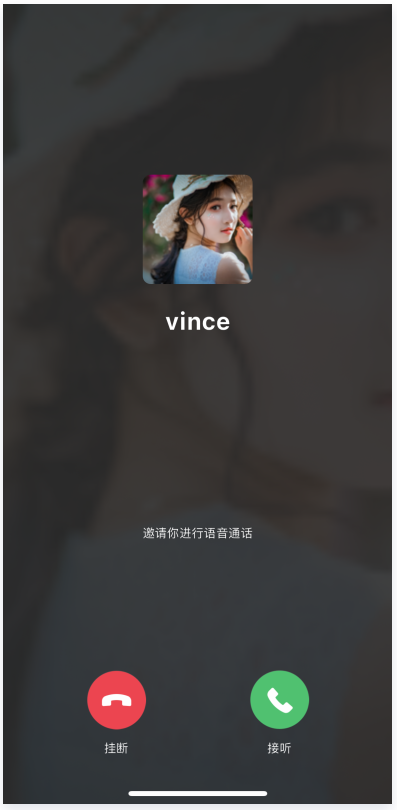
步骤六：拨打您的第一通电话

主叫方与被叫方登录成功后，主叫方通过调用 TUICallKit 的 call 方法并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话，被叫方此时就可接受到来电邀请。

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

void call() {
    TUICallKit.instance.call('vince', TUICallMediaType.audio);
}
```



主叫方



被叫方

更多特性

- 设置昵称、头像
- 界面定制
- 离线推送
- 群组通话
- 悬浮窗
- 美颜特效
- 自定义铃声
- 监听通话状态
- 云端录制

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 。

交流与反馈

- 如果您在使用过程中，有什么建议或者意见，可以在这里反馈：TUICallKit 产品反馈问卷，感谢您的反馈。
- 如果您是开发者，也欢迎您加入我们的 TUICallKit 技术交流平台 zhiliao，进行技术交流和产品沟通。