实时互动-教育版

开发指南

腾讯云

【联系我们】

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

文档目录

开发指南
开发流程概述

最近更新时间：2026-03-18 16:41:02

您可以按照以下步骤进行开发与集成：

步骤	开发阶段	操作指引
1	**接入准备**	完成 **接入准备**，获取应用 ID 等开发必备参数。
2	**快速跑通**	参考 **创建并进入课堂** 中的各平台指南，快速实现基础的上课功能。 **注：**若开发移动端 APP，建议使用原生内核 SDK，详见 **移动端原生内核与 Web 内核的区别** 。
3	**定制开发**	成功跑通首堂课程后，参考 **定制化开发** 打造专属的课堂界面与交互。
4	**进阶参考**	查阅 **进阶功能** 进行更深入的业务集成。

开发者支持与反馈

在开发过程中，如果您遇到任何技术难题或 API 使用疑问，欢迎通过 联系我们 加入官方开发者微信群。我们的技术与产品专家将在线为您答疑解惑。

接入准备

最近更新时间：2026-03-18 16:41:02

本文档将指导您完成互动课堂（LCIC）的接入准备工作，包括账号注册、应用创建、关键密钥及课堂参数的获取。

步骤 1：账号注册与认证

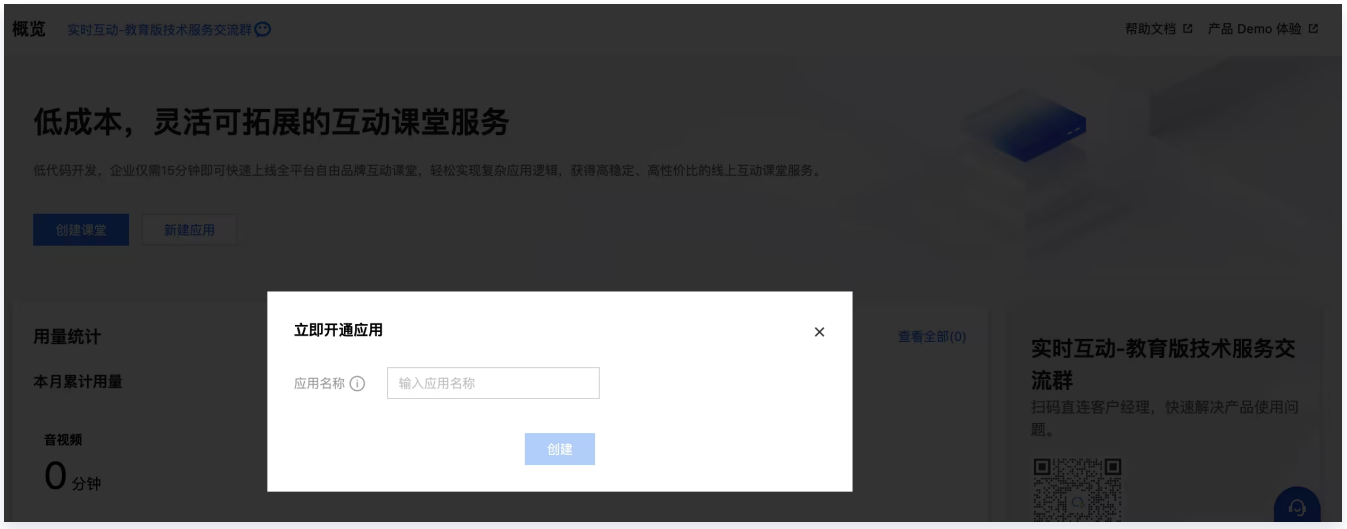
在使用腾讯云服务之前，请确保您已拥有腾讯云账号并完成实名认证。

1. **注册账号**：前往 腾讯云官网 注册账号。
2. **实名认证**：登录后，请完成 实名认证，以便正常使用云服务。

步骤 2：创建应用

登录控制台并创建一个新的互动课堂应用实例。

1. 登录 实时互动-教育版控制台 。
2. 在左侧导航栏中选择**概览**，单击页面中的**新建应用**。
3. 在弹出的“创建应用”界面中，输入您的**应用名称**（例如 TestLCIC），并确认创建。



> **说明：**
> - 应用名称仅支持下划线（_）、数字或中英文字符。
> - 每个账号可免费领取一个**试用版应用**。如需商用，请前往 购买页 根据业务需求创建对应版本的应用。

步骤 3：获取应用配置与密钥

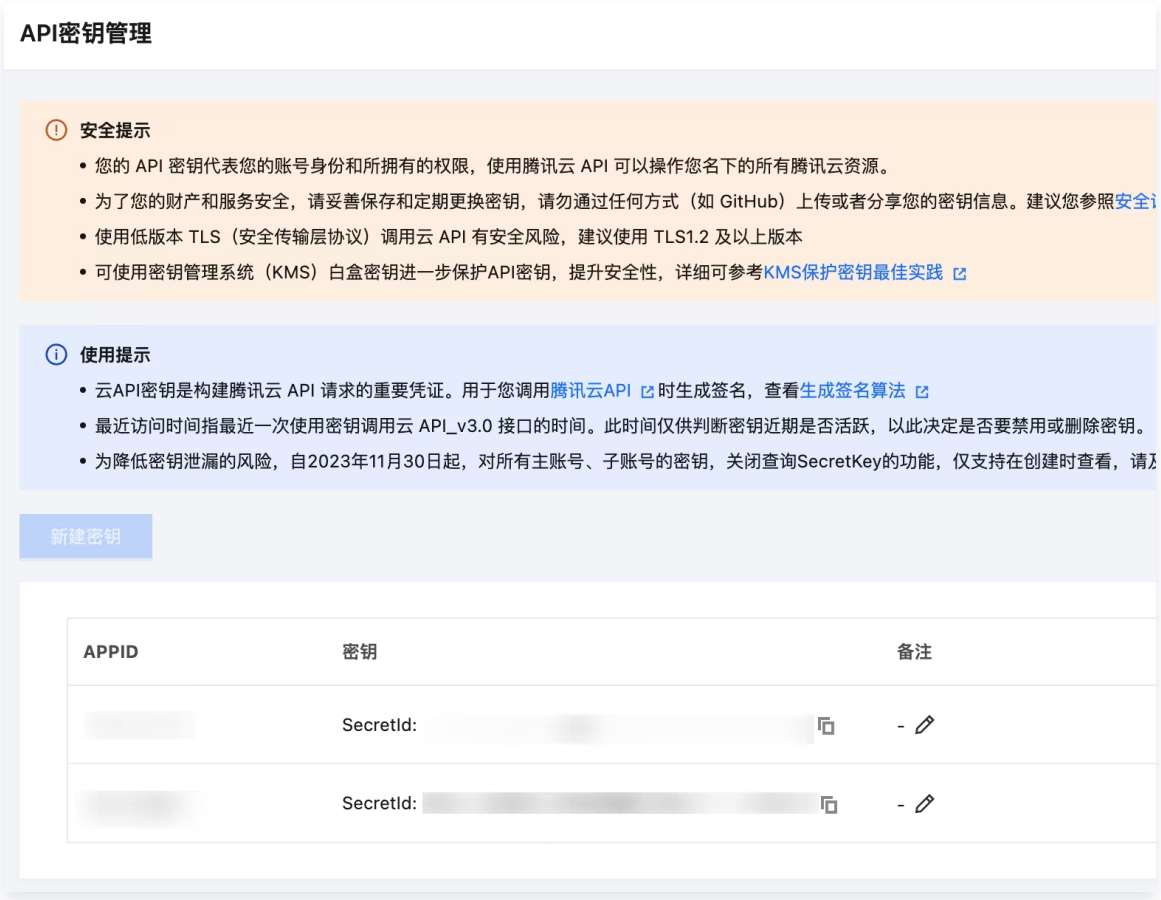
开发过程中需要使用 SDKAppId 和 SecretKey 进行鉴权和配置，请按照以下步骤获取。

3.1 获取 SDKAppId

1. 进入 应用管理 页面。
2. 在页面中找到并复制您应用的**应用id（ SDKAppId）**。

3.2 获取 API 密钥 (SecretKey)

1. 进入 访问管理 (CAM) 控制台 。
2. 在 API 密钥管理列表中查看您的 SecretId 和 SecretKey(密钥)。
   若当前无密钥，请单击**新建密钥**生成。详细操作指引请参见 访问密钥管理 。



步骤 4：获取课堂参数

进入课堂需要以下几个核心参数，请依次获取：

1. **获取应用 ID（`schoolid`）**

   通过步骤2创建应用，从控制台获取对应的应用 ID（即 `SdkAppId`）。

2. **注册用户（`userid`）**

   调用云 API 接口 RegisterUser 注册用户，获取返回的唯一用户标识 `userid`。

3. **获取鉴权令牌（`token`）**

   调用云 API 接口 LoginUser 登录用户，获取用于身份验证的 `token`。

4. **创建课堂（`classid`）**

   调用云 API 接口 CreateRoom 创建课堂，获取返回的课堂号 `classid`。

Web 无 UI SDK 接入

最近更新时间：2026-06-08 18:02:30

概述

本文档面向希望构建**定制化在线互动课堂产品**的前端开发者，介绍腾讯云实时互动-教育版 **无 UI SDK** 的接入方式。SDK 封装了音视频通信、课堂管理、成员管理、课件白板等底层能力，并以响应式状态驱动数据与事件、灵活简便的方法调用提供能力。
您只需关注 UI 层的实现，无需关注底层业务逻辑，即可快速完成在线互动课堂的集成开发。可以完全从头实现属于自己的课堂页面、交互和课堂流程，实现品牌和体验上的独特感。
您也可以直接参考 完整版文档 了解更多高级用法。

> **AI 辅助开发：**
> 为简化并加速接入，我们还提供 AI Skill 辅助，您可以使用主流 AI 编码工具（Cursor / Claude / Copilot 等），通过 Vibe Coding 方式最快速度完成项目开发。**AI 辅助开发文档**
>
> ```
> npx skills add InteractiveClassroom/skills --skill tcic-sdk-helper
> ```

前提条件

- 已完成 接入准备，了解如何获取 `schoolId`、`classId`、`userId`、`token`。
- Node.js >= 16。
- 支持 ES Module 的现代浏览器（Chrome 85+ / Edge 86+ / Safari 15+ / Firefox 80+）。
- **目前暂仅支持 Web 浏览器项目, Electron 及 移动端 Mobile Native 接入将于后期提供扩展。**

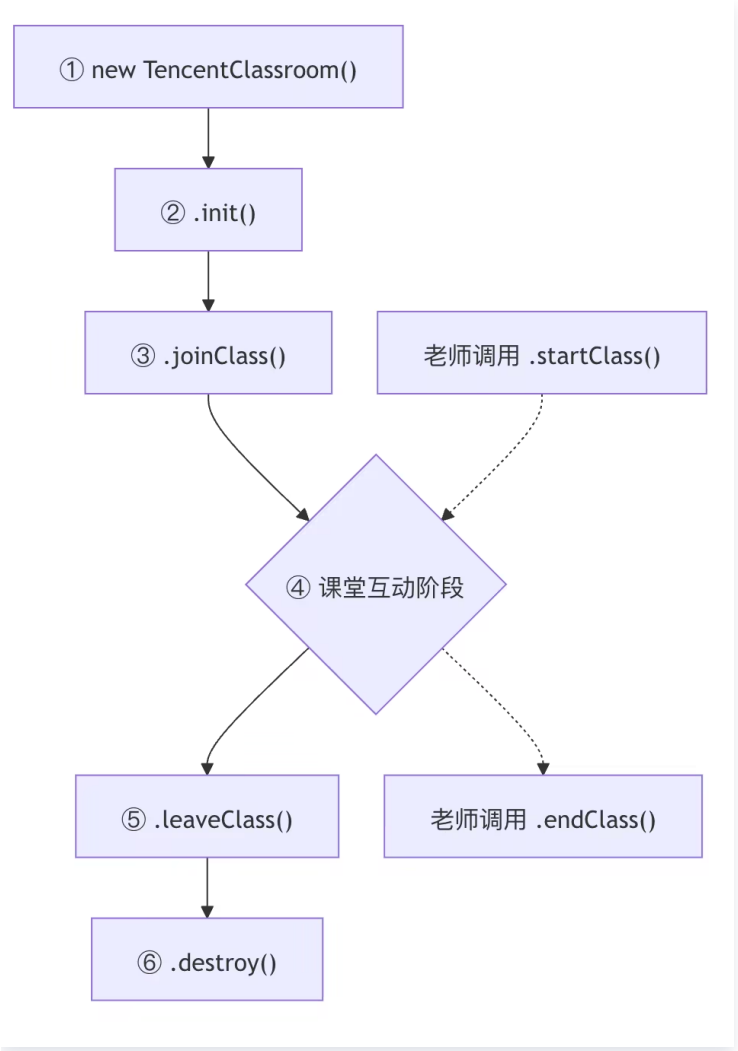
安装

```
npm install @tencent-classroom/sdk
# 或
pnpm add @tencent-classroom/sdk
```

CDN 等更多引入方式请见 安装文档。

课堂生命周期

SDK 的核心调用流程如下：



基本使用流程

步骤1：创建实例

构造方法调用方式

```
import { TencentClassroom, TEvent, StreamType } from '@tencent-
classroom/sdk';

const classroom = new TencentClassroom({
  language: 'zh-CN',
  board: {
    domId: 'board-container', // 白板挂载容器（不需要白板可去掉）
  },
});
```

创建后可通过 TencentClassroom.instance 在任意位置访问实例。

步骤2：初始化

```
const initResult = await classroom.init();
if (!initResult.ok) {
  console.error('初始化失败:', initResult.message);
  return;
}
```

步骤3：注册事件监听

> **说明：**
> 事件必须在 *joinClass()* 之前注册，否则可能丢失进房过程中触发的事件。

```
classroom.on(TEvent.KICK_OUT, ({ message }) => {
  alert(message); // 被踢出课堂
});

classroom.on(TEvent.CLASS_END, () => {
  // 课堂结束
});
```

完整事件列表：全局事件 API

步骤4：加入课堂

调用 *joinClass()* 加入完成加入课堂流程。

```
// classId / userId / token 通过接入准备文档获取
const joinResult = await classroom.joinClass({
  classId: 123456789,
  userId: 'user_001',
  token: 'your-token',
});

if (!joinResult.ok) {
  console.error('进房失败:', joinResult.message);
  return;
}
```

步骤5：音视频推流

调用 startCamera 及 startMicrophone 完成开启本地摄像头及麦克风。

```
// 开启摄像头（传入本地预览 DOM）
await classroom.startCamera(document.getElementById('local-video')!);

// 开启麦克风
await classroom.startMicrophone();
```

> ⚠ **注意：**
> 学生角色必须上台后（ `stageStatus$` = `'active'` ）才能推流。

详细音视频用法： **音视频指南**

步骤6：远端流渲染

监听台上成员变化，并动态为台上成员绑定 DOM 完成展示画面和声音。**完整流程细节可参考详细说明** 。

```
const bindings = new Map();

classroom.state.stageList$.subscribe((stageMembers) => {
  const currentIds = new Set(stageMembers.map(m => m.userId));

  // 解绑已下台成员
  for (const [userId, binding] of bindings) {
    if (!currentIds.has(userId)) {
      classroom.unbindRemoteView(binding);
      bindings.delete(userId);
    }
  }

  // 绑定新上台成员
  for (const member of stageMembers) {
    if (member.userId === myUserId) continue; // 跳过自己
    if (bindings.has(member.userId)) continue;

    const dom = document.getElementById(`video-${member.userId}`);
    const binding = classroom.bindRemoteView(member.userId, dom,
StreamType.Camera);
    bindings.set(member.userId, binding);
```

```
  }
});
```

步骤7：离开课堂

```
// 解绑所有远端流
for (const [, binding] of bindings) {
  classroom.unbindRemoteView(binding);
}

// 离房 + 销毁（两者都必须调用）
await classroom.leaveClass();
await classroom.destroy();
```

各角色接入要点

老师（Teacher）

步骤	操作
加入课堂	`joinClass({ role: 'teacher' })`
开启音视频	`startCamera()` + `startMicrophone()`（自动上台）
开始上课	`startClass()`
课中管理	`muteAudioAll()` / `setSilenceMode()` / `memberAction()`
结束上课	`endClass()`
退出	`leaveClass()` → `destroy()`

学生（Student）

步骤	操作
加入课堂	`joinClass({ role: 'student' })`
等待开课	订阅 `classStatus$`
观看远端流	`bindRemoteView(teacherId, DOM, StreamType.Camera)`
举手连麦	`handUp()` → 等待老师同意

上台推流	`stageStatus$` = `'active'` 后调用 `startCamera()`
退出	`leaveClass()` → `destroy()`

助教（Assistant）

- 与老师相同管理权限，默认在台上。
- 可直接开启音视频，可执行成员管理操作。

巡课（Supervisor）

- 只读模式：可观看音视频、查看白板、浏览花名册。
- 不可发消息、不可操作成员、不出现在成员列表中。

互动消息聊天

```
// 发送文本消息
await classroom.sendTextMessage('Hello');

// 订阅消息列表
classroom.state.messageList$.subscribe((messages) => {
  renderChatList(messages);
});
```

更多消息类型（图片/文件/自定义消息/撤回/私聊）及更多用法：**互动消息聊天指南**。

互动白板及课件

在构造实例时配置白板容器，进房后自动初始化：

```
const classroom = new TencentClassroom({
  board: {
    domId: 'board-container',
    ratio: '16:9',
  },
});
```

白板工具操作、翻页、课件管理等：**白板及课件指南**。

国际化

SDK 内置 9 种语言支持：

```
const classroom = new TencentClassroom({
  language: 'en', // zh-CN | zh-TW | en | ja | ko | vi | ar | id | es
});
```

详细用法（运行时切换/词条覆盖/新增语言）： **国际化指南** 。

完整示例

```
import { TencentClassroom, TEvent, StreamType } from '@tencent-
classroom/sdk';

async function main() {
  const classroom = new TencentClassroom({
    language: 'zh-CN',
    board: { domId: 'board-container' },
  });

  const initResult = await classroom.init();
  if (!initResult.ok) return console.error(initResult.message);

  // 注册事件（进房前）
  classroom.on(TEvent.KICK_OUT, ({ message }) => alert(message));

  // 进房（参数通过接入准备文档获取）
  const joinResult = await classroom.joinClass({
    classId: 123456789,
    userId: 'user_001',
    token: 'your-token',
  });
  if (!joinResult.ok) return console.error(joinResult.message);

  // 开启音视频
  await classroom.startCamera(document.getElementById('local-video')!);
  await classroom.startMicrophone();

  // 渲染远端流
  const binding = classroom.bindRemoteView(
    'teacher_001',
    document.getElementById('remote-video')!,
    StreamType.Camera,
```

```
  );

  // 页面卸载时清理
  window.addEventListener('beforeunload', async () => {
    classroom.unbindRemoteView(binding);
    await classroom.leaveClass();
    await classroom.destroy();
  });
}

main();
```

打包部署

开发完成后，您可以将项目编译打包并部署到自己的 Web 服务器上运行。
更多细节及高级用法，**请参考完整版文档** 。

开发者支持与反馈

在开发过程中，如果您遇到任何技术难题或 API 使用疑问，欢迎通过 **联系我们** 加入官方开发者微信群。我们的技术与产品专家将在线为您答疑解惑。

创建并进入课堂
Web 及 H5 快速接入

最近更新时间：2026-06-08 18:02:30

本文档将指导您如何通过服务端 API 创建课堂，并生成 Web 端访问链接以进入课堂。

> **说明：**
> 本方案适用于，您希望直接使用我们提供的课堂 UI 前端使用（可参考 Demo），且定制化程度不大的场景。
> 如果您的目标效果和前述我们提供的 UI 差异比较大，**建议您使用 无 UI SDK 进行深度定制**，基于我们提供的互动课堂底层能力，仅需简单关注页面 UI，即可快速完成课堂项目搭建，不需要关注业务逻辑及数据处理。可以完全从头实现属于自己的课堂页面、交互和课堂流程，实现品牌和体验上的独特感。

进入课堂

参考 接入准备 获取必要参数后，通过拼接 URL 即可访问互动课堂网页版。
请将获取到的参数替换至以下 URL 模板中（可参考 参数详解）：

```
https://${请根据下述表格选用课堂站点域名}/latest/class.html?
schoolid=${schoolid}&classid=${classid}&userid=${userid}&token=${token}
```

> **注意：**
> 请对 URL 中的参数值进行 encodeURIComponent 编码处理。

课堂站点域名

请根据用户所在地域及实际情况，选用一个合适的课堂站点域名，供用户访问。拼接到上述的课堂进房链接中。

域名	服务器地域	说明
class.qcloudclass.com	中国大陆（全球加速）	中国大陆用户及全球不确定位置用户选用。
www.tencentclass.com	新加坡（全球加速）	非中国大陆用户首选访问，海外主体独立域名且纯海外原生部署，性能优化，全球可用性高。

iframe 集成

如果您需要将课堂页面集成到 iframe 中，需要给 iframe 元素添加 `allow` 属性，以确保课堂页面可以正确获取到所需的浏览器权限。

```
<iframe
    allow="camera; microphone; fullscreen; display-capture; clipboard-
read; clipboard-write; autoplay;"
    src="https://${请根据上方表格选用课堂站点域名}/latest/class.html?
userid=12345&token=xxx"
>
```

参数详解

核心鉴权参数（必填）

以下参数为进入课堂的必要条件，缺失将导致无法正常访问。

参数字段	类型	含义	说明
schoolid	String	应用 ID	在 **接入准备** 中创建的应用 ID（`SdkAppId`）。
userid	String	用户 ID	通过 **RegisterUser** 接口获取。
token	String	鉴权 Token	通过 **LoginUser** 接口获取。
classid	String	课堂 ID	通过 **CreateRoom** 接口获取。

基础配置参数（可选）

用于控制课堂的语言、角色及场景模式。

参数	类型	默认值	说明
lng	String	`zh-CN`	界面语言设置。支持值： • `zh-CN`：中文简体 • `zh-TW`：中文繁体 • `en-US`：英语 • `ja`：日语 • `ko`：韩语 • `ar`：阿拉伯语 • `vi`：越南语

			• `id`：印尼语
role	String	空	进入课堂的角色。可选值： `supervisor`：巡课/内容审查（仅限已注册应用内 巡课用户 ）。
scene	String	`default`	场景名称，用于区分不同的定制布局，有两种配置方式： 方式1：使用 SetAppCustomContent 接口配置。 方式2：控制台 场景配置 。

设备与行为控制（可选）

用于控制用户进入课堂时的设备状态及交互行为。

参数字段	类型	默认值	说明
micAutoOpen	Number	–	麦克风默认状态（仅对上台用户有效）。 • `0`：关闭 • `1`：开启
cameraAutoOpen	Number	–	摄像头默认状态（仅对上台用户有效）。 • `0`：关闭 • `1`：开启
location	Boolean	`false`	是否上报经纬度位置信息。
noEndClass	Boolean	`false`	若设置该参数（任意值），将隐藏「下课」，仅展示「离开」按钮。
back_url	String	空	退出或返回课堂时的回调跳转地址。 **注意：该参数值必须使用 `encodeURIComponent` 进行 URL 编码。**

UI 与布局定制（可选）

用于自定义课堂的视觉表现。

参数字段	类型	默认值	说明
layout	String	`top`	布局模式（仅SubType为videodoc时有效）。 支持值： • `top`：顶部布局（默认） • `double`：双排布局

			• `right`：右侧布局 • `left`：左侧布局 • `three`：三分布局
boardColor	String	`#182E25`	白板背景颜色。支持 Hex 格式或 rgba(0, 0, 0, .3) 格式。
debugjs	String	-	自定义 UI 的 JS 文件链接。本地开发时使用，仅支持 localhost。
debugcss	String	-	自定义 UI 的 CSS 文件链接。本地开发时使用，仅支持 localhost。

Android 快速接入
原生内核（Android）快速接入

最近更新时间：2026-04-27 11:02:11

本文将介绍如何在 Android 项目中快速集成互动课堂功能。

> **说明：**
> 阅读本文前，您可以先了解 移动端原生内核与 Web 内核的区别 。

开发环境要求

在开始之前，请确保您的开发环境满足以下要求：

- **Android SDK**：API 21 及以上 (Android 5.0+)
- **NDK**：27.0.12077973 及以上
- **Java**：Java 11 及以上
- **Kotlin**：支持

快速接入 (Quick Start)

以下步骤将介绍如何快速在项目中跑通互动课堂。完整示例代码可参考 官方 Demo 。

步骤 1：导入 SDK

1. 在项目根目录的 `settings.gradle` 中配置 Maven 仓库：

```
maven { url "https://storage.googleapis.com/download.flutter.io" }
maven { url "https://android.qcloudclass.com/repo" }
```

2. 在 app 模块的 `build.gradle` 中添加依赖（请将 `${latest}` 替换为 Demo 中的最新版本号 ）：

```
android {
    compileSdk = 35 // 确保 compileSdk 满足要求
}

dependencies {
    implementation("com.qcloudclass:tcic:${latest}")
}
```

3. 在 `AndroidManifest.xml` 的 `<application>` 标签中添加 `tools:replace="android:allowBackup"` 解决冲突：

```
<application
    android:allowBackup="true"
    tools:replace="android:allowBackup"
    ... >
```

步骤 2：配置权限

在 `AndroidManifest.xml` 中声明必要的硬件和网络权限：

```
<!-- 网络与状态 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<!-- 音视频与蓝牙 -->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"
/>
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<!-- 存储 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

> ⚠ **注意：**
> Android 6.0 以上系统，麦克风、相机、存储等敏感权限需在代码中动态申请

步骤 3：初始化 SDK

在应用启动或首次进入相关页面时进行初始化（只需调用一次）：

```
// Java / Kotlin
TCICManager.initialize(context);
```

步骤 4：设置全局回调监听 (Kotlin 示例)

通过设置回调，您可以监听用户进出房间、收到消息等事件：

```
TCICManager.setCallback(object : TCICManager.TCICCallback {
    override fun onJoinedClassSuccess() {
        runOnUiThread { Toast.makeText(ctx, "加入课堂成功",
Toast.LENGTH_SHORT).show() }
    }

    override fun afterExitedClass() {
        runOnUiThread {
            Toast.makeText(ctx, "已退出课堂", Toast.LENGTH_SHORT).show()
            TCICManager.closeTCICActivity() // 关闭课堂页面
        }
    }

    override fun onJoinedClassFailed() {
        runOnUiThread { TCICManager.closeTCICActivity() }
    }

    override fun onKickedOffClass() {
        runOnUiThread { Toast.makeText(ctx, "被踢出课堂",
Toast.LENGTH_SHORT).show() }
    }

    override fun onMemberJoinedClass(data: Map<*, *>) { /* 成员加入 */ }
    override fun onMemberLeaveClass(data: Map<*, *>) { /* 成员离开 */ }
    override fun onRecivedMessage(message: Map<*, *>) { /* 收到消息 */ }
    override fun onError(errorCode: String, errorMsg: String) { /* 错误处理
*/ }
})
```

步骤 5：进入课堂页面 (Jetpack Compose 示例)

通过 接入准备 拿到关键参数，配置 `TCICConfig` 并进入课堂：

```
@Composable
fun EnterClassRoomButton(context: Context) {
    Button(onClick = {
        // 1. 填入从后台获取的参数
        val token = "your_token"
        val classId = "your_class_id"
```

```
        val userId = "your_user_id"
        val role = 1 // 角色：0学生, 1老师, 3助教, 4巡课

        // 2. 设置配置
        val config = TCICConfig(token, classId, userId, role)
        TCICManager.setConfig(config)

        // 3. 跳转到课堂页面
        context.startActivity(TCICManager.getTCICIntent(context))
    }) {
        Text(text = "点击进入课堂")
    }
}
```

Web 内核（Android）快速接入

最近更新时间：2026-05-09 22:01:51

本文档旨在指导开发者快速将互动课堂（LCIC）Web 内核版 SDK 集成至 Android 项目中。

> **说明：**
> 阅读本文前，您可以先了解 移动端原生内核与 Web 内核的区别 。

开发环境要求

在开始集成前，请确保您的开发环境满足以下要求：

- **开发工具**：Android Studio 3.0 或更高版本。
- **系统版本**：Android 6.0（API Level 23）及以上。

集成步骤

步骤 1：导入 SDK

LCIC SDK 已发布至 Maven Central，建议通过 Gradle 进行远程集成。
请在应用模块（Module）的 `build.gradle` 文件中进行如下配置。

1. 添加依赖

在 `dependencies` 闭包中添加 SDK 依赖：

```
dependencies {
    // LCICSDK 组件
    implementation 'com.tencent.edu:TCICSDK:1.8.32'
}
```

2. 配置 CPU 架构

在 `defaultConfig` 闭包中指定支持的架构。

> **说明：**
> 目前 SDK 支持 `armeabi-v7a` 和 `arm64-v8a` 。您可以根据业务需求灵活配置，支持的架构越多，安装包体积越大。

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

```
    }
}
```

3. 配置 Java 编译版本

在 `compileOptions` 闭包中，设置 JDK 版本为1.8：

```
compileOptions {
    sourceCompatibility 1.8
    targetCompatibility 1.8
}
```

4. 适配高版本 AGP

如果您的 Android Gradle Plugin (AGP) 版本**大于4.2.0**，请在 `android` 闭包中添加以下配置以避免打包错误：

```
android {
    packagingOptions {
        jniLibs {
            useLegacyPackaging true
        }
    }
}
```

5. 同步项目

配置完成后，单击 Android Studio 右上角的 **Sync Now**，等待 SDK 下载并集成完成。

步骤 2：配置清单文件

在 `AndroidManifest.xml` 中声明 SDK 所需的权限：

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

<application
    ...
    <!-- 注意：当 minSdkVersion ≥ 23 时，必须添加此属性 -->
    android:extractNativeLibs="true">
    ...
```

```
</application>
```

步骤 3：配置混淆规则（可选）

如果您开启了代码混淆，请在 `proguard-rules.pro` 文件中添加以下规则，防止 LCIC SDK 核心类被混淆：

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

步骤 4：申请 SDK 授权 (License)

App 开发需要获取以下两种 License：

1. Web 引擎 License

- **测试环境**：请参考 申请 Web 引擎测试用 License，**完成流程至第四部分获取 LicenseKey 即可**。
- **正式环境**：购买旗舰版后，请通过 联系我们 加入微信群联系技术支持申请正式版 License。

2. 播放器授权（仅“直播大班课”需要）

如果是“直播大班课”班型，需申请播放器权限。请按以下模板填写信息并提交 腾讯云工单，相关疑问可通过 联系我们 加入微信群了解。

> ⚠ **注意：**
> 一个旗舰版仅支持授权一个正式包名，请确认信息无误。

参考模板提供信息：

- 问题：实时互动-教育版申请播放器授权
- 公司名称：
- 联系方式：
- App Name：
- Package Name（Android）：
- Bundle ID（iOS）：

步骤 5：初始化 Web 引擎

Web 引擎相较于系统 WebView 具有更好的兼容性和速度。

1. 隐私合规检查

请务必在用户**同意隐私政策协议之后**，再调用初始化方法，以免违规收集个人信息导致应用上架失败。

2. 执行初始化

初始化代码必须在**主进程**中执行，且只需调用一次。

```
// 判断是否为主进程
if (TCICInitProvider.isMainProcess(context)) {
    // 初始化 Web 引擎
    // licenseKey 获取方式请参考步骤 4
    TCICManager.getInstance().initX5Core(licenseKey, new
TBSSdkManageCallback() {
        @Override
        public void onCoreInitFinished() {
            // 内核初始化结束回调
        }

        @Override
        public void onViewInitFinished(boolean isX5Core) {
            // Web 引擎初始化完成，此时可以进入课堂
            Log.i("LCIC", "Web Engine Init Finished. Is X5 Core: " +
isX5Core);
        }
    });
}
```

步骤 6：构建进房参数

请按照文档 接入准备 获取必要参数。
进入课堂需要构建 `TCICClassConfig` 对象。请通过以下方式获取所需参数：

参数详情表

字段	类型	必填	含义	备注
schoolid	int	是	应用 ID	在 接入准备 中创建的应用 ID（`SdkAppId`）。
userid	long	是	课堂 ID	通过 RegisterUser 接口获取。
token	string	是	用户 ID	通过 LoginUser 接口获取。
classid	string	是	鉴权 Token	通过 CreateRoom 接口获取。

region	string	否	服务器地域	默认值 cn，即访问中国大陆服务器节点. 支持值： • cn：中国大陆。国内用户或全球不确定位置用户。 • sg：新加坡。全球非中国大陆用户。
scene	string	否	场景名称	用于区分不同的定制布局，有两种配置方式： 方式1：使用 SetAppCustomContent 接口配置。 方式2：控制台 场景配置 。
lng	string	否	语言参数	界面语言设置。支持值： • `zh-CN`：中文简体 • `zh-TW`：中文繁体 • `en-US`：英语 • `ja`：日语 • `ko`：韩语 • `ar`：阿拉伯语 • `vi`：越南语 • `id`：印尼语 需配合 `TCICWebViewManager.getInstance().setClassLanguage` 使用。
camera	int	否	开启摄像头	`1`：开启 (默认)，`0`：关闭。
mic	int	否	开启麦克风	`1`：开启 (默认)，`0`：关闭。
speaker	int	否	开启扬声器	`1`：开启 (默认)，`0`：关闭。
groupLiveCode	string	否	分组直播码	详见 分组直播 。

步骤 7：调起课堂页面

使用配置好的参数启动 `TCICClassActivity` 即可进入课堂。

```
// 1. 构建配置对象
TCICClassConfig initConfig = new TCICClassConfig.Builder()
    .schoolId(schoolId)   // 应用 ID
    .classId(classId)     // 课堂 ID
    .userId(userId)       // 用户 ID
    .token(token)         // 鉴权 Token
    .build();
```

```
// 2. 启动课堂 Activity
Intent intent = new Intent(getActivity(), TCICClassActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
Bundle bundle = new Bundle();
bundle.putParcelable(TCICConstants.KEY_INIT_CONFIG, initConfig);
intent.putExtras(bundle);
startActivity(intent);
```

监听退出课堂

如果您需要感知用户退出课堂的事件，可以注册本地广播接收器：

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(TCICConstants.ON_CLASS_EXITED_ACTION);
LocalBroadcastManager.getInstance(context).registerReceiver(broadcastRecei
ver, intentFilter);
```

推荐与资源

- **异常监控**：建议接入 腾讯 Bugly 以快速发现并解决线上异常，提升用户体验。
- **示例代码**：请参考 Android 开发 Demo 。
- **进阶功能**：如需定制界面，请参考 Web 内核定制指南 。

iOS 快速接入
原生内核（iOS）快速接入

最近更新时间：2026-03-18 16:41:02

本文档将指导您如何在 iOS 项目中快速接入腾讯云互动课堂 SDK。

> **说明：**
> 阅读本文前，您可以先了解 移动端原生内核与 Web 内核的区别 。

开发环境要求

- **IDE**：Xcode 14 及以上版本
- **系统要求**：iOS 11.0 及以上

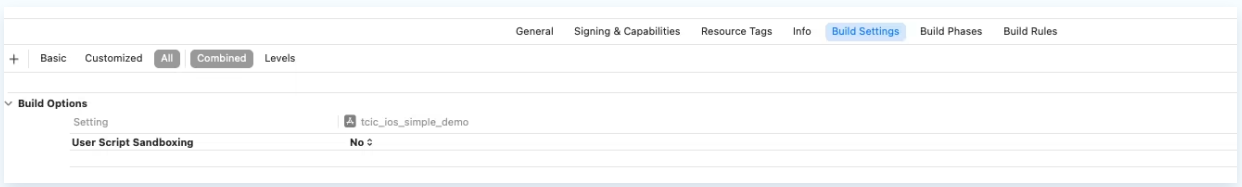
集成步骤

按照以下步骤，您可以快速在 iOS 项目中集成并跑通互动课堂。完整示例代码可参考 官方 Demo 。

步骤一：安装依赖

在项目的 `Podfile` 中添加 SDK 依赖：

```
# framework 使用静态集成
use_frameworks! :linkage => :static

pod 'tcic_ios', :podspec =>
'https://ios.qcloudclass.com/${latest}/tcic_ios.podspec'
```

> **说明：**
> - 如果没有 `Podfile`，请在项目根目录执行 `pod init` 生成。
> - `${latest}` 版本号请参考 Demo Podfile（最新版本会同步更新至 Demo）。
> - 请在 Xcode 的 Build Settings 中将 `User Script Sandboxing` 设置为 `No`。
>
> 
>
>
> - 请务必使用真机进行调试，音视频等硬件相关插件在模拟器上无法正常运行。

步骤二：权限配置

教学场景通常需要使用音视频互动和屏幕共享功能。请在项目的 `Info.plist` 文件中添加以下权限描述：

```
<!-- 相册权限 -->
<key>NSPhotoLibraryUsageDescription</key>
<string>Video calls require photo library permission.</string>

<!-- 相机权限 -->
<key>NSCameraUsageDescription</key>
<string>Video calls require camera permission.</string>

<!-- 麦克风权限 -->
<key>NSMicrophoneUsageDescription</key>
<string>Voice calls require microphone permission.</string>

<!-- 网络配置 -->
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>

<!-- 后台模式 -->
<key>UIBackgroundModes</key>
<array>
    <string>fetch</string>
    <string>processing</string>
</array>

<!-- 屏幕录制扩展 -->
<key>com.apple.security.application-groups</key>
<array>
    <string>group.com.tencent.comm.tcic.sharescreen</string>
</array>
```

步骤三：初始化 SDK

在应用启动或首次进入相关页面时，需要初始化 TCIC SDK。

```
import SwiftUI
```

```
import tcic_ios

@main
struct tcic_ios_simple_demoApp: App {
    init() {
        // 初始化 SDK
        TCICManager.shared.initialize()
    }

    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

步骤四：设置回调事件

通过设置回调，您可以监听课堂内的各类事件（如进退房、消息等）。全局只需设置一次。

```
@State private var callback: TCICCallback = TCICCallback() // 初始化回调事件

self.callback.afterExitedClassBlock = {
    print("dismiss page")
}
self.callback.onJoinedClassFailedBlock = {
    print("joined class failed")
}

// 注册回调
TCICManager.shared.setCallback(callback)
```

步骤五：配置课堂参数

通过 接入准备 拿到关键参数。在进入课堂前，需构建 `TCICConfig` 对象并传入必要参数。

```
let headerConfig = TCICHeaderComponentConfig()
let messageConfig = TCICMessageComponentConfig()

// 示例：配置自定义头部左侧视图
// headerConfig.headerLeftBuilder = headerLeftBuilder
```

```
let config = TCICConfig(
    token: "YOUR_TOKEN",        // 通过云 API 获取的 token
    classId: "YOUR_CLASS_ID",   // 课堂 ID
    userId: "YOUR_USER_ID",     // 用户 ID
    role: 1,                    // 用户角色 (0: 学生, 1: 老师, 3: 助教, 4: 巡
课)
    headerComponentConfig: headerConfig,    // Header 组件配置
    messageComponentConfig: messageConfig   // Message 组件配置
    // ... 其他自定义配置
)

TCICManager.shared.setConfig(config)
```

步骤六：进入课堂页面

在合适的时机（如按钮点击事件中）拉起课堂页面。以下为 SwiftUI 示例：

```
Button(action: {
    let headerLeftBuilder: TCICHeaderComponentConfig.HeaderBuilder = {
        return MyHeaderLeftView(messenger:
TCICManager.shared.Tengine.binaryMessenger)
    }

    let headerConfig = TCICHeaderComponentConfig()
    let messageConfig = TCICMessageComponentConfig()
    headerConfig.headerLeftBuilder = headerLeftBuilder
    headerConfig.headerLeftBuilderWidth = 200
    headerConfig.headerLeftBuilderHeight = 40

    let config = TCICConfig(
        token: "YOUR_TOKEN",
        classId: "YOUR_CLASS_ID",
        userId: "YOUR_USER_ID",
        role: 1,
        headerComponentConfig: headerConfig,
        messageComponentConfig: messageConfig
    )

    TCICManager.shared.setConfig(config)
    isActive = true
```

```
}) {
    Text("打开互动课堂")
        .frame(maxWidth: .infinity)
        .padding()
        .background(Color.blue)
        .foregroundColor(.white)
        .cornerRadius(8)
}
.padding(.horizontal)
.fullScreenCover(isPresented: $isActive, onDismiss: {
    TCICManager.shared.Tengine.viewController = nil
}) {
    TCICManager.TPage()
        .edgesIgnoringSafeArea(.all)
}
```

Web 内核（iOS）快速接入

最近更新时间：2026-05-09 22:01:51

本文档旨在指导 iOS 开发者快速将互动课堂（LCIC）SDK 集成至现有项目中。

> **说明：**
> 阅读本文前，您可以先了解 移动端原生内核与 Web 内核的区别 。

开发环境准备

在开始集成前，请确保您的开发环境满足以下要求：

- **开发工具**：Xcode 14 或更高版本。
- **系统要求**：iOS 11.0 或更高版本。

集成步骤

步骤 1：导入 SDK

LCIC SDK 已发布至 CocoaPods。请在项目的 `Podfile` 中添加以下依赖，并执行 `pod install` 进行安装。

```
pod 'TCICSDK_Pro', '1.8.5.20'
pod 'TXLiteAVSDK_Professional', '12.7.19324'
```

步骤 2：配置 App 权限

LCIC SDK 需要使用相机、麦克风及相册权限。请在主 App 的 `Info.plist` 文件中添加以下键值对，并填入相应的权限申请描述：

```
<key>NSCameraUsageDescription</key>
<string>需要访问您的相机以进行视频互动</string>
<key>NSMicrophoneUsageDescription</key>
<string>需要访问您的麦克风以进行语音互动</string>
<key>NSPhotoLibraryAddUsageDescription</key>
<string>需要访问相册以保存图片</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>需要访问相册以读取图片</string>
```

步骤 3：全局环境配置

在启动课堂前，建议进行全局环境参数配置（如域名、语言等）。

```
// 1. 设置课堂域名
// 默认域名为：class.qcloudclass.com
// targetDomain 必须为腾讯提供的专用域名
[TCICClassController setDomain:targetDomain];

// 2. 设置 H5 端版本号
// 默认为："latest"。一般情况下无需调整
[TCICClassController setH5Version:targetVersion];

// 3. 设置课堂语言类型
// 默认为中文："zh"。具体参数值请参考下文参数表
[TCICClassController setClassLanuage:[msgDic objectForKey:@"lng"]];

// 4. 预加载资源
// 注意：预加载逻辑需放在最后调用，前面的配置变更会清除预加载内容
[TCICClassController preloadClass];
```

步骤 4：准备课堂参数

请按照文档 接入准备 获取必要参数。
进入课堂需要构建 `TCICClassConfig` 对象。请通过以下方式获取所需参数：

参数详情表

字段	类型	必填	含义	备注
schoolid	int	是	应用 ID	在 接入准备 中创建的应用 ID（`SdkAppId`）。
userid	long	是	课堂 ID	通过 RegisterUser 接口获取。
token	string	是	用户 ID	通过 LoginUser 接口获取。
classid	string	是	鉴权 Token	通过 CreateRoom 接口获取。
region	string	否	服务器地域	默认值 cn，即访问中国大陆服务器节点。支持值： • cn：中国大陆。国内用户或全球不确定位置用户。 • sg：新加坡。全球非中国大陆用户。

scene	string	否	场景名称	用于区分不同的定制布局，有两种配置方式： 方式1：使用 SetAppCustomContent 接口配置。 方式2：控制台 场景配置 。
lng	string	否	语言参数	界面语言设置。支持值： • zh-CN：中文简体 • zh-TW：中文繁体 • en-US：英语 • ja：日语 • ko：韩语 • ar：阿拉伯语 • vi：越南语 • id：印尼语 需配合 TCICWebViewManager.getInstance().setClassLanguage 使用。
camera	int	否	开启摄像头	1：开启 (默认)，0：关闭。
mic	int	否	开启麦克风	1：开启 (默认)，0：关闭。
speaker	int	否	开启扬声器	1：开启 (默认)，0：关闭。
groupLiveCode	string	否	分组直播码	详见 分组直播 。

步骤 5：调起课堂页面

构建 `TCICClassConfig` 对象并传入必要参数，即可调起 LCIC 组件主页面。

打开课堂代码示例：

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];

// 必填参数
roomConfig.schoolId = 123456;          // 学校/应用 ID
roomConfig.userId = "test_user";     // 用户 ID
roomConfig.token = "test_token";     // 鉴权 Token
roomConfig.classId = 654321;          // 课堂 ID

// 可选参数配置
```

```
[roomConfig setValue:@"en" forKey:@"language"];          // 设置语言
[roomConfig setValue:@"scene_name" forKey:@"scene"];     // 设置场景
[roomConfig setValue:@(0) forKey:@"preferPortrait"];     // 屏幕方向：0 横屏（默认），1 竖屏

// 初始化并跳转控制器
TCICClassController *vc = [TCICClassController classRoomWithConfig:roomConfig];
if (vc) {
    [(UINavigationController *)self.window.rootViewController pushViewController:vc animated:YES];
} else {
    NSLog(@"错误：参数配置有误，无法初始化课堂控制器");
}
```

监听退出通知：

如果您需要监听用户退出课堂的事件，请注册 `TCICExitClassRoomCompleteNotify` 通知：

```
[[NSNotificationCenter defaultCenter] addObserver:self

selector:@selector(handleClassExit)

name:@"TCICExitClassRoomCompleteNotify"
                                          object:nil];
```

步骤 6：SDK 授权申请

如果是“直播大班课”班型，需申请播放器权限。请按以下模板填写信息并提交 腾讯云工单，相关疑问可通过 联系我们 加入微信群了解。

参考模板提供信息：

- 问题：实时互动-教育版申请播放器授权
- 公司名称：
- 联系方式：
- App Name：
- Package Name（Android）：
- Bundle ID（iOS）：

> ⚠ **注意：**
> **一个旗舰版仅支持授权一个正式包名，请确认信息无误。**

高级功能：移动端屏幕分享

LCIC SDK 支持 iOS 系统级屏幕分享功能。实现该功能需要利用 iOS 的 ReplayKit 框架及 App Group 能力。

1. 创建 App Group

请参考 TRTC 官网文档 > 步骤 1：创建 App Group 完成 App Group 的创建。

2. 创建 Broadcast Upload Extension

请参考 TRTC 官网文档 > 步骤 2：创建 Broadcast Upload Extension 在工程中创建一个新的 Target（类型为 Broadcast Upload Extension）。

3. 配置 Extension 依赖

在 `Podfile` 中为新创建的 Extension Target 添加依赖，并重新执行 `pod install`。

```
target 'YourExtensionTargetName' do
    # 如果不使用动态库，请注释下一行
    # use_frameworks!
    pod 'TCICSDK_Pro_ReplayKit'
end
```

4. 实现 SampleHandler

在 Extension 的 `SampleHandler.m` 文件中，导入头文件并实现以下逻辑。

> ⚠ **注意：**
> 请将 `APPGROUP` 宏定义的值修改为您在 上文 中创建的 App Group ID。

```
#import "SampleHandler.h"
#import <TXLiteAVSDK_ReplayKitExt/TXLiteAVSDK_ReplayKitExt.h>
#import <TCICScreenKit/TCICScreenKit.h>

// TODO: 替换为您创建的 App Group Identifier
#define APPGROUP "group.com.yourcompany.app"

@interface SampleHandler() <TXReplayKitExtDelegate>
@end

@implementation SampleHandler
```

```
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject *>
*)setupInfo {
    // 初始化 ReplayKitExt
    [[TXReplayKitExt sharedInstance] setupWithAppGroup:APPGROUP
delegate:self];
    // 通知 LCIC ScreenKit 开始
    [[TCICScreenKit sharedScreenKit] onScreenKitStarted];
}

- (void)broadcastPaused {
    [[TCICScreenKit sharedScreenKit] onScreenKitPaused];
}

- (void)broadcastResumed {
    [[TCICScreenKit sharedScreenKit] onScreenKitResumed];
}

- (void)broadcastFinished {
    [[TXReplayKitExt sharedInstance] finishBroadcast];
    [[TCICScreenKit sharedScreenKit] onScreenKitFinished];
}

#pragma mark - TXReplayKitExtDelegate

- (void)broadcastFinished:(TXReplayKitExt *)broadcast reason:
(TXReplayKitExtReason)reason {
    NSString *tip = @"";
    switch (reason) {
        case TXReplayKitExtReasonRequestedByMain:
            tip = @"屏幕共享已结束";
            break;
        case TXReplayKitExtReasonDisconnected:
            tip = @"应用断开";
            break;
        case TXReplayKitExtReasonVersionMismatch:
            tip = @"集成错误（SDK 版本号不匹配）";
            break;
    }
    NSError *error = [NSError
errorWithDomain:NSStringFromClass(self.class) code:0 userInfo:@{
        NSLocalizedFailureReasonErrorKey:tip
```

```
    }];
    [self finishBroadcastWithError:error];
}

- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:
(RPSampleBufferType)sampleBufferType {
    // 假设 kSupportSceenShare 是您控制是否开启的宏或变量，如不需要可移除判断
    if (kSupportSceenShare) {
        switch (sampleBufferType) {
            case RPSampleBufferTypeVideo:
            case RPSampleBufferTypeAudioApp:
                // 分发数据给 LCIC ScreenKit
                [[TCICScreenKit sharedScreenKit]
processSampleBuffer:sampleBuffer withType:sampleBufferType];
                // 分发数据给 TXLiteAVSDK
                [[TXReplayKitExt sharedInstance]
sendSampleBuffer:sampleBuffer withType:sampleBufferType];
                break;
            case RPSampleBufferTypeAudioMic:
                // 麦克风音频通常由主 App 采集，此处一般忽略
                break;
            default:
                break;
        }
    }
}
@end
```

5. 主 App 对接配置

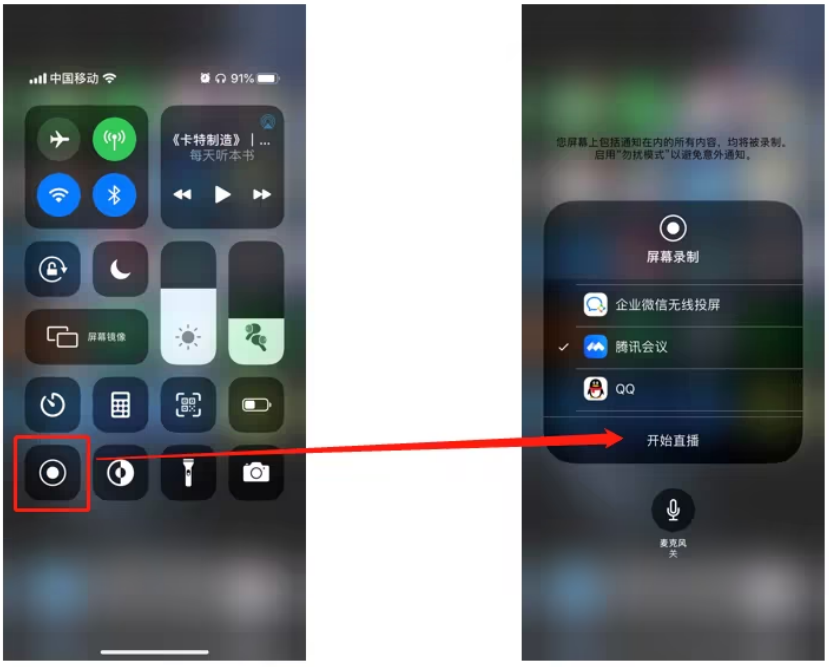
在主 App 初始化课堂配置时，必须设置 `appGroup` 参数，以便主 App 能与 Extension 通信。

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];
roomConfig.userId = "test";
roomConfig.token = "test_token";
roomConfig.classId = 123454;
roomConfig.schoolId = 123456; // 同 SDKAppId

// 【关键步骤】通过 KVC 设置 AppGroup
[roomConfig setValue:@"group.com.yourcompany.app" forKey:@"appGroup"];
```

6. 注意事项

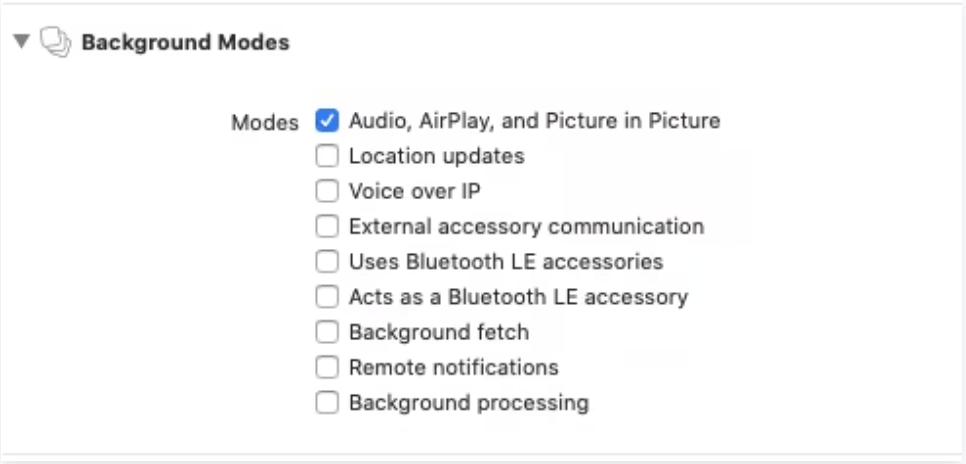
1. **触发方式：**
   - **iOS 12及以上**：TCICSDK 已内置屏幕分享触发按钮（需不依赖 Scene 生命周期）。具体可参见 TRTC 官网文档 > 步骤 4：增加屏幕分享的触发按钮（可选），但该功能有限制条件。
     如果代码中已支持 `Scenedelegate`，可参见 Xcode 11 删除 Scenedelegate，进行移除。
     以 Demo 为例，弹出效果如下，单击**开始直播**即可。

   - **iOS 11**：需引导用户在控制中心长按录屏按钮，选择您的 Extension 应用（如“腾讯会议”示例图所示）来启动。



长按“录屏” 按钮　　点击“开始直播”

2. **版本兼容**：Extension 的 `Deployment Target` 建议设置为 iOS 11.0。
3. **后台模式**：主 App 必须开启 Background Modes 中的 “Audio, AirPlay, and Picture in Picture” 选项，以保证后台保活。



附录与参考

- **异常监控**：建议接入 腾讯 Bugly 以监控应用稳定性。
- **示例代码**：下载 iOS 开发 Demo 参考完整实现。
- **进阶配置**：更多自定义页面配置请参考 Web 内核定制指南 。

Electron 快速接入

最近更新时间：2026-05-09 22:01:51

本文档旨在指导开发者如何在 Electron 环境下快速接入互动课堂（LCIC）。
根据业务需求，我们提供 **URL 唤起** 和 **SDK 集成** 两种主要的接入方式，并提供客户端定制与 UI 自定义的高级功能说明。

接入方式

方式一：URL 唤起

使用浏览器或 Web 页面通过自定义协议（Scheme）唤起本地客户端。

浏览器直接唤起

1. **下载客户端**：前往 Demo 下载及体验 下载并安装 Windows 或 Mac 客户端（旗舰版支持定制 OEM 包）。
2. **唤起链接**：当浏览器访问以下 URL 时，将自动请求打开客户端：
   参数可以参考 Web 及 H5 快速接入 的参数说明。

```
tcic://${选用一个合适的课堂站点域名从下方表格}/latest/class.html?
classid=${classId}&userid=${userId}&token=${token}
```

课堂站点域名

请根据用户所在地域及实际情况，选用一个合适的课堂站点域名，供用户访问。拼接到上述的课堂进房链接中。

域名	服务器地域	说明
class.qcloudclass.com	中国大陆（全球加速）	中国大陆用户及全球不确定位置用户选用。
www.tencentclass.com	新加坡（全球加速）	非中国大陆用户首选访问，海外主体独立域名且纯海外原生部署，性能优化，全球可用性高。

中转页唤起（最佳实践）

建议业务侧实现一个“中转页”，通过集成 `ElectronProtocolCheck.js` 优化用户体验：自动检测客户端是否安装，成功则直接唤起，失败则引导下载，或降级至 Web 版课堂。
集成步骤：

1. **下载检测库**：下载 ElectronProtocolCheck.js 并引入项目。
2. **实现唤起逻辑**：在中转页中获取上课参数（ `classId`，`userId`，`token` ），调用检测函数。

示例代码：

```
// 引入检测库（代码实现参考 GitHub 示例）
import ElectronProtocolCheck from './ElectronProtocolCheck';

// 1. 准备入参
const classId  = 123456;
const userId = "JIUzI1NiIsIn123456";
// 注意：Token 需从后台动态获取，避免过期
const token = 'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...';

// 2. 拼接唤起 URL
const url = `tcic://${选用一个合适的课堂站点域名从上方表格}/latest/class.html?classid=${classId}&userid=${userId}&token=${token}`;
console.log(`callClient->start: ${url}`);

// 3. 执行唤起与回调处理
ElectronProtocolCheck(
  url,
  () => {
    // 成功回调：客户端已唤起
    console.log('callClient->success!');
  },
  () => {
    // 失败回调：未检测到客户端
    console.log('callClient->failed!');
    // 建议逻辑：弹出提示窗口，引导用户下载客户端
    // 若用户取消下载，可降级跳转至 Web/H5 版本课堂
  },
  () => {
    // 不支持回调：浏览器环境不支持自定义协议
    // 建议逻辑：直接加载 Web 版课堂链接
  }
);
```

方式二：SDK 集成

将互动课堂嵌入到您现有的 Electron 应用中。我们提供了 Electron Demo 供参考。

环境要求

请确保您的开发环境满足以下基础库版本：

开发框架	版本要求
Node.js	16.14.2

安装 SDK

在项目根目录下执行 npm 命令安装 SDK：

```
npm install tcic-electron-sdk@latest
```

> **说明：**
> 最新版本信息请查看 npm: tcic-electron-sdk 。

初始化与进课

引入模块并调用初始化接口，传入鉴权参数即可调起课堂窗口。
请根据用户所在地域及实际情况，选用一个合适的课堂站点域名，供用户访问。拼接到上述的课堂进房链接中。

域名	服务器地域	说明
class.qcloudclass.com	中国大陆（全球加速）	中国大陆用户及全球不确定位置用户选用。
www.tencentclass.com	新加坡（全球加速）	非中国大陆用户首选访问，海外主体独立域名且纯海外原生部署，性能优化，全球可用性高。

```
const TCIC = require('tcic-electron-sdk');

TCIC.initialize({
  // 核心鉴权参数
  classId: '368507569',
  userId: '123456',
  token: 'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...', // 务必动态获取

  // 可选：完整 URL（若提供，优先级高于 classId 等参数）
  url: 'https://${选用一个合适的课堂站点域名从上方表格}/latest/class.html?
classid=xxxx&userid=xxx',

  // 可选：自定义参数（透传至 Web 端）
```

```
  customParams: { key: 'value' },

  // 可选：业务标识
  sign: 'your_signature',
  cid: 'customer_id',
  uid: 'user_uid',

  // 生命周期回调
  onReady() {
    console.log('课堂窗口正在拉起');
  },
  afterInit() {
    console.log('SDK 初始化完成');
  },
  onClose() {
    console.log('所有课堂窗口已关闭');
  }
});
```

高级功能

1. OEM 定制打包

若您需要拥有独立品牌（Logo、应用名称）的客户端，并希望快速集成现有业务系统，可申请使用“OEM 定制打包”服务。

接入流程

1. 准备资料发送给腾讯云（商务或者产品经理），或通过 联系我们 加入微信群申请。

> **说明：**
> **仅支持已购买旗舰版的客户申请。**

字段	说明
公司名称	您所在的公司名称。
腾讯云账号 UIN	腾讯云账号 ID。在腾讯云 账号信息 查看“账号ID”。
应用 ID	您创建的应用 ID。可参考 接入准备 创建并获取应用 ID。
AppName	App 名称。例如：XX 课堂。

Logo	应用图标。 • 规格：256 × 256 • 格式：ico/png。
URL	业务入口 URL。客户端启动后默认加载的页面地址。 业务 URL 必须具备自动处理登录态的能力（或自动跳转至登录页）。请勿直接配置为登录页，以免用户每次打开客户端都需要重新登录。

在接收到资料后，并确认无误的情况下我们将会在1个工作日完成打包。

2. **业务逻辑适配：**

在打包后，业务侧可通过 `window.joinClass` 方法进入课堂，若之前业务侧已通过 URL 或 SDK 集成方式实现进课逻辑，可参考以下示例进行兼容。

```
const options = {
  classId: '368507569',
  userId: '123456',
  token: '...'
};

// 检测是否在定制客户端环境中
if (window.joinClass) {
  // 调用定制客户端的进课方法
  window.joinClass(options);
} else {
  // 原有的 Web/SDK 进课逻辑
}

// 退出客户端逻辑
// window.closeWin();
```

2. 自定义 UI 集成

支持开发者自定义课中布局及样式。通过注入 JS 和 CSS，您可以修改界面元素。

调试阶段

在开发调试阶段，可通过 URL 参数 `debugjs` 和 `debugcss` 注入本地或远程的资源文件。

- **URL 拼接方式：**

```
tcic://...?
classid=...&debugjs=http://localhost:443/myLib.js&debugcss=http://localh
```

```
ost:443/myLib.css
```

- **SDK 集成方式：**

```
TCIC.initialize({
  // ...其他参数
  debugjs: 'http://localhost:443/demo/dist/myLib.umd.min.js',
  debugcss: 'http://localhost:443/demo/dist/myLib.css',
})
```

开发

详细的自定义 JS 开发方式可参考 Web 内核定制指南 。

生产发布

调试完成后，请勿继续使用 debug 参数。请通过以下任意方式将自定义资源绑定到“场景”中：

1. 方式一：使用云 API SetAppCustomContent 。
2. 方式二：前往 控制台 > **应用管理** > **应用配置** > **场景配置**进行绑定。

进课堂时，只需在 URL 或 SDK 参数中指定 `scene` 参数，即可加载对应的 UI 布局。

小程序快速接入

最近更新时间：2026-03-18 16:41:02

本文档旨在指导开发者如何将**实时互动-教育版 (LCIC)** 小程序插件集成至现有的小程序项目中。该插件提供完整的互动课堂与直播功能。

接入前准备

在开始集成前，请确保您满足以下要求。

1. 资质与类目要求

本插件属于小程序的**社交 > 直播**类目，**仅限国内注册的非个人主体**（如企业、政府、媒体等）使用。
请确保您的小程序符合以下类目要求，并在 微信公众平台 完成类目设置。

一级类目	二级类目
电商平台	电商平台
	网上竞价平台（文物）
	网上竞价平台（非文物）
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：**

- **主体要求：**必须为非个人主体，否则无法使用直播组件。
- **类目审核：**请在小程序后台**设置 > 基本设置 > 服务类目**中配置。所选类目需与实际业务场景一致，否则审核将被驳回。以上类目表格仅提供参考，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。
- **主体一致性：**小程序主体无需与腾讯云账号主体一致。

2. 环境要求

请确保开发环境满足以下最低版本要求：

- **微信 App iOS**：7.0.9 及以上
- **微信 App Android**：7.0.8 及以上
- **小程序基础库**：2.10.0 及以上

集成步骤

步骤1：申请插件

登录微信公众平台，在**设置 > 第三方设置 > 插件管理**中添加插件：

- **插件名称**：实时互动教育版
- **AppID**：`wxbc2aedd3838d78cd`
- 也可以直接单击 申请链接 。

> **说明：**
> 请确保您的小程序类目符合要求，否则可能搜索不到、添加不了。

步骤2：引入插件

1. 在小程序全局配置文件 `app.json` 中声明插件：

```
{
  "plugins": {
    ...
    "tcic-plugin": {
      "version": "1.8.0",
      "provider": "wxbc2aedd3838d78cd",
      "export": "tcicPluginConfig.js",
      "genericsImplementation": {
        "interactive": {
          "interactive-class-info": "components/live/interactive-class-
info"
        }
      }
    }
    ...
  }
}
```

2. 在 `app.json` 同级目录下创建 `tcicPluginConfig.js`，用于配置插件行为及导出方法：
   或者放置其他目录也可以，但插件配置的 export 字段也需要进行相应的变更，参考官方文档 导出到插件 。

```
// tcicPluginConfig.js
const app = getApp();

module.exports = {
  setValue: app.setValue, // 关键：用于传递 IM 消息等事件
  loginUrl: '/pages/index/login', // 业务侧登录页路径
  homeUrl: '/pages/index/home',    // 业务侧首页路径
  Config: {
    LIVE: {
      MAX_TEXT_NUM: 200, // IM 文本最大长度
      RECORD_TIPS: '您可以到课程列表中查看回放',
    },
    IMG: {
      MAX_SIZE: 20 * 1024 * 1024, // 图片上传限制 20MB
    },
    ENABLE_TROPHY: true, // 启用奖励特效
    HIDE_USER_LIST_BAR: false, // 是否隐藏用户列表
    ORIENTATION: 'portrait', // 屏幕方向：portrait(竖屏) / landscape(横屏)
    NAV_TITLE: '实时互动课堂', // 导航栏标题
    VIDEO_WATER_MARK: '', // 视频水印 URL
  },
};
```

3. 在 `app.js` 中实现 `setValue` 方法，用于处理插件与宿主小程序的通信：

```
// app.js
const events = new Map();

App({
  onLaunch() {},

  // 插件通过此方法与小程序通信
  setValue(key, data) {
    const value = events.get(key);
    if (Array.isArray(value)) {
      value.forEach((e) => {
        if (typeof e.callback === 'function') {
          e.callback(data);
        }
      });
```

```
    }
  }
});
```

步骤 3：跳转进入课堂

请按照文档 接入准备 获取必要参数。
在业务页面通过 `navigator` 组件或 API 跳转至插件页面。

使用 API 跳转

推荐使用 API 跳转。

```
goToLiveRoom() {
  const params = {
    classid: '222',        // 课堂 ID
    userid: 'user_001',    // 用户 ID
    token: 'token_str',    // 鉴权 Token
    type: 'interactive'    // interactive(互动课) 或 live(直播课)
  };

  const queryString = Object.keys(params)
    .map(key => `${key}=${params[key]}`)
    .join('&');

  wx.navigateTo({
    url: `plugin://tcic-plugin/${params.type}?${queryString}`
  });
}
```

使用 navigator 组件

使用 navigator 组件。

```
<navigator url="plugin://tcic-plugin/interactive?
classid=123&userid=abc&token=xyz&type=interactive">
  进入课堂
```

```
</navigator>
```

参数说明

进入课堂时需传入以下参数，请根据业务需求生成：

字段	类型	必填	说明	备注
userid	String	是	用户 ID	来自 `RegisterUser` 接口。
classid	String	是	课堂 ID	来自 `CreateRoom` 接口。
token	String	是	鉴权 Token	来自 `LoginUser` 接口。
type	String	是	课堂类型	可选值： • interactive：互动小班课、强互动大班课 • live：直播大班课

步骤 4：业务组件开发

若需自定义直播间简介、水印或分享配置，需自行实现相关组件（如 `class-info` 、`landscape-im-container` 等）。

参考代码：请查看 微信小程序代码片段 。需额外注意的地方：

1. app.js 中挂载 setValue,setWatching,unsetWatching。
2. utils 上挂载 formatTimeStamp。
3. 缓存参数：

```
wx.setStorageSync('classid', info.classid);
wx.setStorageSync('userid', info.userid);
wx.setStorageSync('token', info.token);
wx.setStorageSync('classmode', info.type);  // interactive
wx.setStorageSync('nickname', info.username);
```

4. 关键代码：
   - 直播简介

     components/live/class-info.js 修改 getClassInfo。
   - 分享内容配置

components/live/class-info.js 修改 tcicPlugin.setShareInfo。

- 直播画面水印配置

  components/live/class-info.js 修改 tcicPlugin.setValue('watermark', {...})。

5. 配置 Request 合法域名：

```
https://tcic-api.qcloudclass.com;
https://tcic-demo.qcloudclass.com;
https://tcic-api.qcloudtiw.com;
https://tcic-demo.qcloudtiw.com;
https://tcic-demo-login.qcloudclass.com;
```

常见问题 (FAQ)

有问题可通过 联系我们 加群咨询。

插件体积较大，导致主包超出 2M 限制怎么办?

建议将插件接入放在**分包**中。

1. 在 `app.json` 的 `subpackages` 中配置分包。
2. 从分包页面跳转至插件页。

> ⚠ **注意：**
> 若遇到 `tcicPluginConfig.js is not defined` 错误，请将该配置文件复制一份到分包的根目录下。

分包中引入插件，样式不生效?

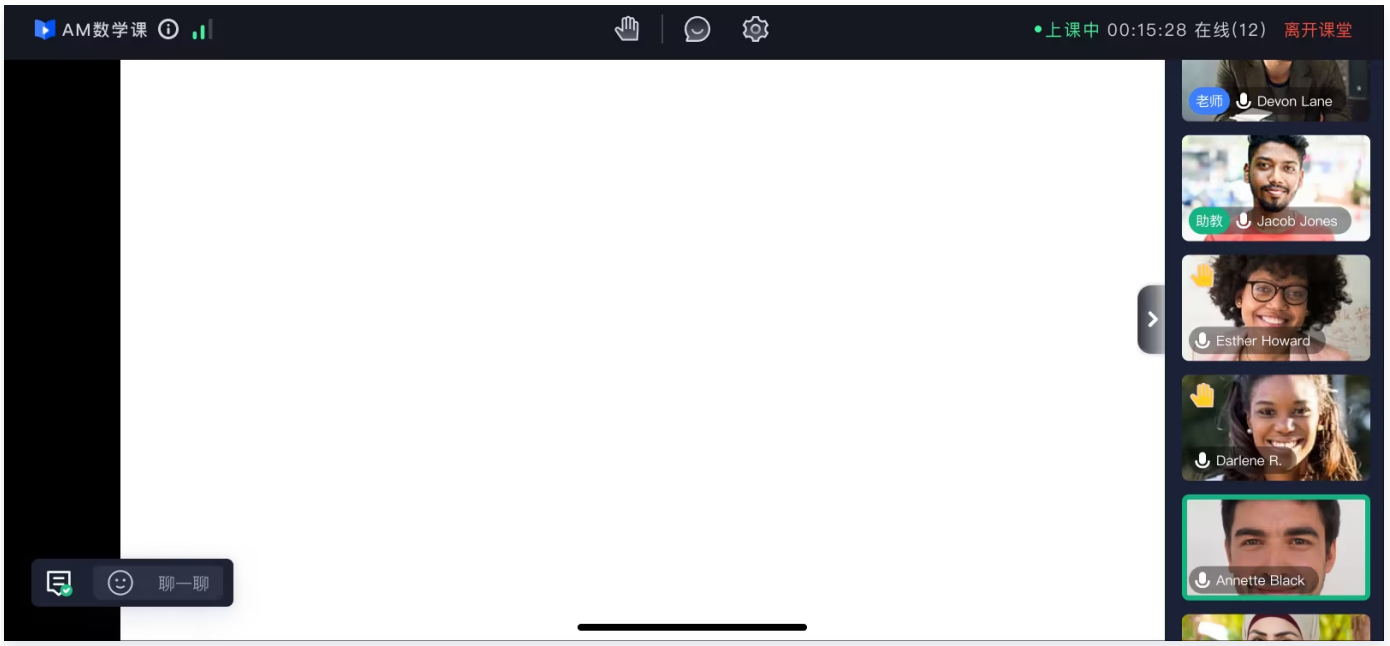
这是微信开发工具的已知问题。
解决方案：将分包所需的公共样式写入 `app.wxss` 中作为全局样式。详情可参考 微信开放社区反馈 。

Flutter 快速接入

最近更新时间：2026-03-18 16:41:02

腾讯云实时互动-教育版是一款集成音视频连麦、互动白板和直播等多功能的产品，能够帮助您节省90%的开发工作量。在教育、医疗、金融和企业培训等领域，可帮助您快速搭建一对一教学、互动小班课、直播大班课和直播带货等多种互动直播业务场景。

`TCIC Client` 是一个包含 LCIC 所有功能的 Flutter 插件，您可以通过该插件快速集成 LCIC 功能，实现互动直播、互动白板、音视频连麦等核心能力。



支持平台

- **Android**：API 21+ (Android 5.0 及以上)
- **iOS**：12.0+

环境要求

Flutter 环境

- **Flutter SDK**：^3.7.0
- **Dart SDK**：⩾ 1.17.0

Android 环境

- **Android SDK**：API 21+
- **NDK**：27.0.12077973+
- **Java**：11+
- **Kotlin**：支持

iOS 环境

- **iOS**：12.0+
- **Xcode**：14.0+
- **CocoaPods**：1.12.0+

> **说明：**
> 在线 DEMO 体验，可单击查看 TCIC Flutter Simple Demo 。

快速跑通

1. 安装依赖

在您的 Flutter 项目根目录下执行以下命令安装依赖：

```
flutter pub add tcic_client_ui
```

2. 权限相关配置

在教学场景中，通常需要使用音视频互动、屏幕共享等功能，因此需在原生工程中配置对应权限。

Android 端配置

1. 权限配置

在 `android/app/src/main/AndroidManifest.xml` 中添加以下权限：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"
/>
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

2. 屏幕录制配置

同样在 `AndroidManifest.xml` 中添加屏幕录制辅助 Activity 及画中画支持：

```
<activity
    android:name=".MainActivity"
    android:supportsPictureInPicture="true" /> <!-- 画中画功能 -->

<activity

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

3. build.gradle 配置

在 `android/app/build.gradle.kts` 中进行如下配置：

```
android {
    namespace = "com.tencent.tcic"
    compileSdk = flutter.compileSdkVersion
    ndkVersion = "27.0.12077973" // 指定 NDK 版本

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }

    kotlinOptions {
        jvmTarget = JavaVersion.VERSION_11.toString()
    }

    defaultConfig {
        applicationId = "com.tencent.tcic"
        minSdk = flutter.minSdkVersion
        targetSdk = flutter.targetSdkVersion
    }

    // 解决重复 so 文件冲突
    packagingOptions {
        pickFirst("lib/arm64-v8a/libliteavsdk.so")
        pickFirst("lib/armeabi-v7a/libliteavsdk.so")
        pickFirst("lib/x86/libliteavsdk.so")
        pickFirst("lib/x86_64/libliteavsdk.so")
```

```
    }
}
```

4. 混淆配置

在 `android/app/proguard-rules.pro` 中添加：

```
# 腾讯云 SDK 混淆配置
-keep class com.tencent.** { *; }
-keep class com.tencent.rtc.** { *; }
-keep class com.tencent.liteav.** { *; }
-keep class com.tencent.trtc.** { *; }
-keep class com.tencent.imsdk.** { *; }
-keep class com.tencent.tls.** { *; }
-dontwarn com.tencentcloudapi.cls.android.producer.**

# Flutter 相关
-keep class io.flutter.** { *; }
-keep class io.flutter.plugins.** { *; }
```

iOS 端配置

权限配置

在 `ios/Runner/Info.plist` 中添加以下权限描述：

```
<!-- 相机权限 -->
<key>NSCameraUsageDescription</key>
<string>互动课堂需要访问您的相机以进行视频通话。</string>

<!-- 麦克风权限 -->
<key>NSMicrophoneUsageDescription</key>
<string>互动课堂需要访问您的麦克风以进行语音通话。</string>

<!-- 网络配置 -->
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>

<!-- 后台模式 -->
```

```
<key>UIBackgroundModes</key>
<array>
    <string>audio</string>
    <string>fetch</string>
    <string>picture-in-picture</string>
    <string>processing</string>
</array>

<!-- 屏幕录制扩展 App Group -->
<key>com.apple.security.application-groups</key>
<array>
    <string>group.com.tencent.comm.tcic.sharescreen</string>
</array>
```

3. 跑通代码

配置完权限后，将通过 接入准备 中获取的 `classId` 、 `userId` 、 `token` 传入组件即可启动课堂。

```
import 'package:tcic_client_ui/tcic_client_ui.dart';
import 'package:tcic_client_ui/controller/tcic_contoller.dart';
import 'package:tcic_client_ui/utils/model/enum/role_enum.dart';
import 'package:tcic_client_ui/utils/model/tcic_cofig_model.dart';

// 1. 创建控制器
final controller = TCICController();

// 2. 配置参数
final config = TCICConfig(
  userId: 'user123',         // 替换为您的 userId
  classId: 'class456',       // 替换为您的 classId
  role: RoleEnum.student,    // 设置角色
  token: 'your_token_here', // 替换为您的 token
);

// 3. 跳转到课堂页面
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => TCICView(
      controller: controller,
      config: config,
```

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

跨应用屏幕共享 (iOS 专属)

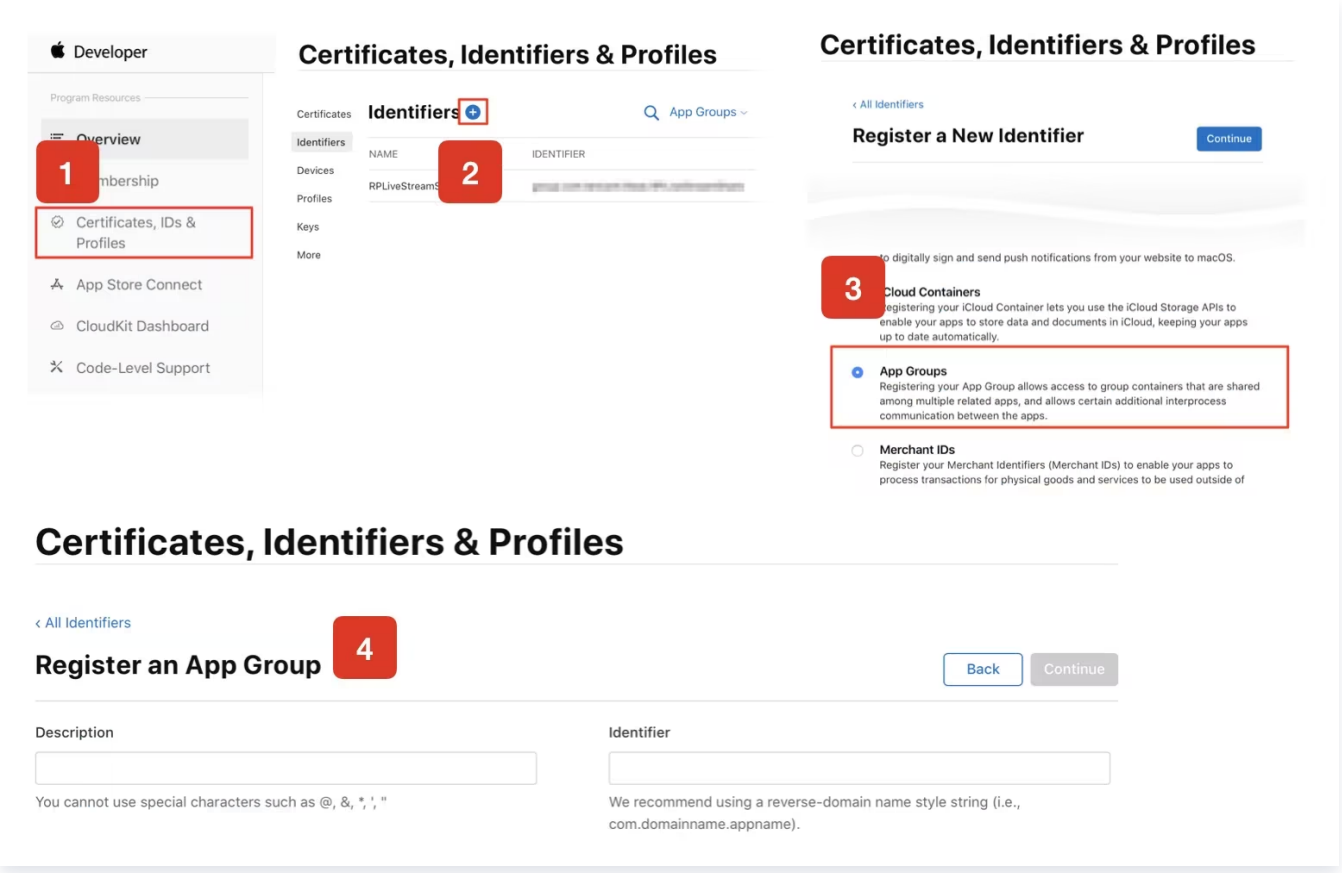
iOS 系统上的跨应用屏幕分享，需要增加 Extension 录屏进程以配合主 App 进程进行推流。Extension 录屏进程由系统在需要录屏时创建，负责接收系统采集到的屏幕图像。

> **注意：**
> - 如果跳过步骤1（即不配置 App Group，接口传 null），屏幕分享依然可以运行，但稳定性会有所降低。
> - 强烈建议配置正确的 App Group 以保障功能稳定性。

步骤1：创建 App Group

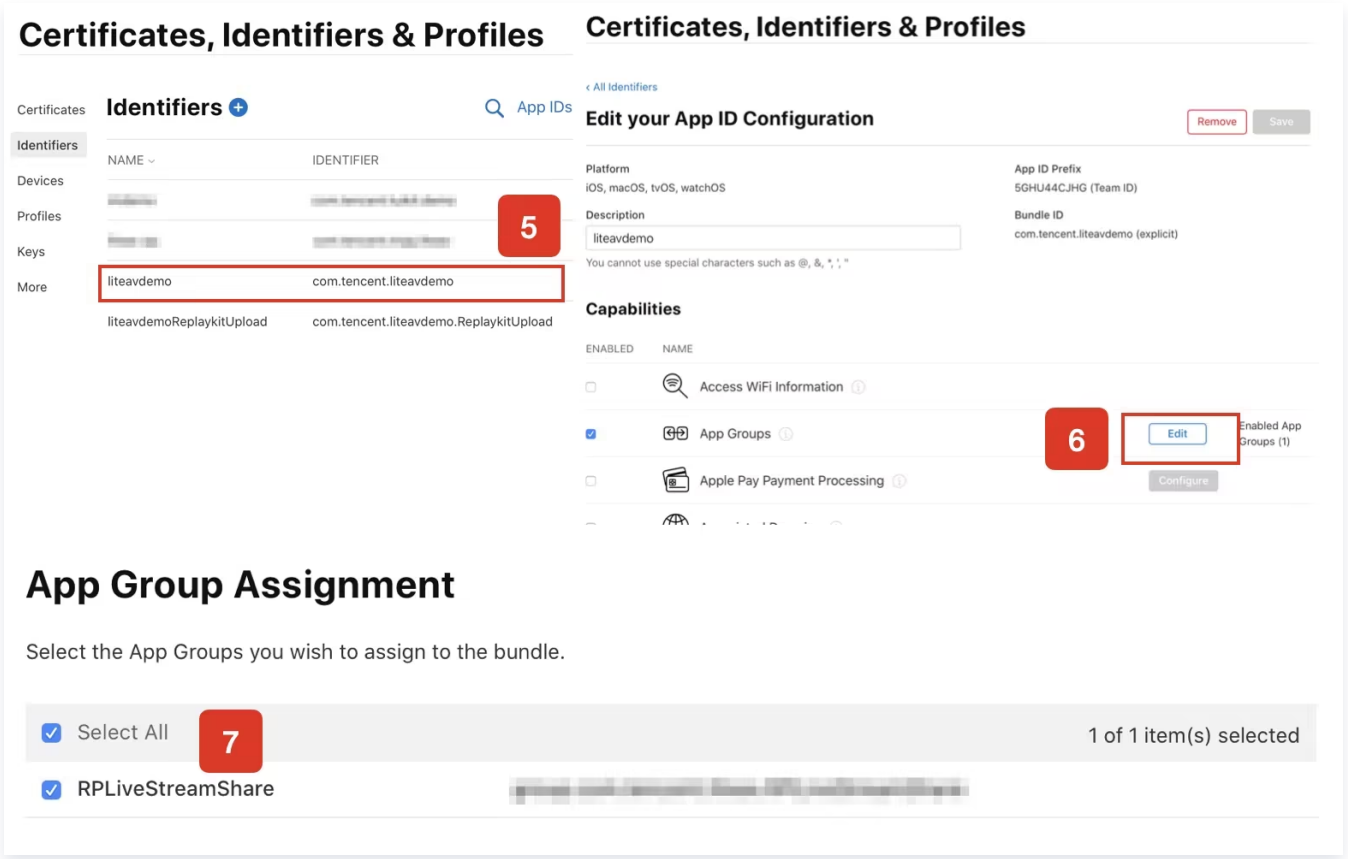
使用您的账号登录 Apple Developer，按以下步骤操作（完成后需重新下载 Provisioning Profile）：

1. 单击左侧的 **Certificates, IDs & Profiles**。
2. 在右侧界面中单击 + 号。
3. 选择 **App Groups**，单击 **Continue**。
4. 填写 Description 和 Identifier（Identifier 需传入接口中的 AppGroup 参数），单击 **Continue**。



5. 回到 Identifier 页面，选择 **App IDs**，单击您的 App ID（主 App 与 Extension 需进行相同配置）。

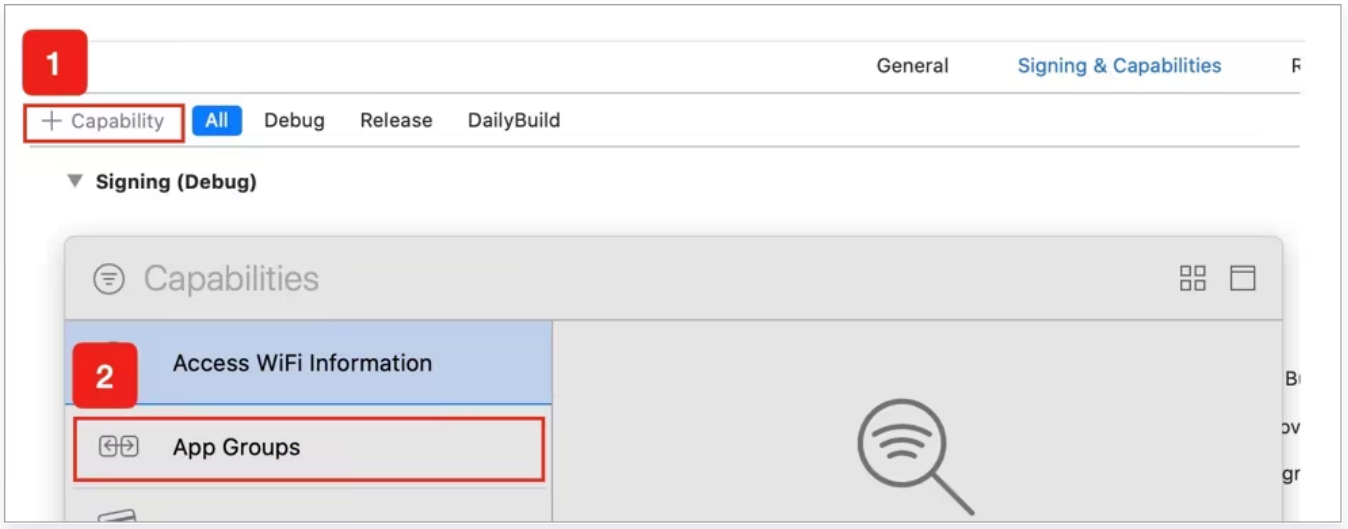
6. 选中 **App Groups** 并单击 **Edit**。
7. 选择刚创建的 App Group，保存配置。



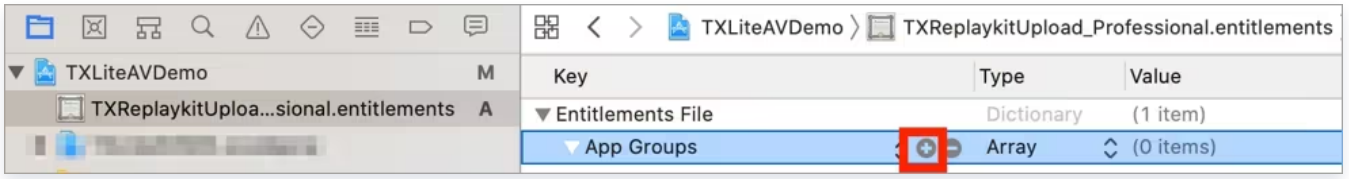
8. 重新下载 Provisioning Profile 并配置到 Xcode 中。

步骤2：创建 Broadcast Upload Extension

1. 在 Xcode 菜单依次单击 **File** > **New** > **Target...**，选择 **Broadcast Upload Extension**。
2. 填写相关信息（**无需勾选** Include UI Extension），单击 **Finish**。
3. 获取定制的 TXLiteAVSDK_ReplayKitExt.xcframework，将其拖入工程并勾选刚创建的 Target。
4. 选中新 Target，单击 **+ Capability**，双击 **App Groups**。



5. 在生成的 `Target.entitlements` 文件中填写上述创建的 App Group Identifier。



6. 对主 App 的 Target 执行相同的 App Group 配置。
7. 在新 Target 中，将自动生成的 `SampleHandler.swift` 文件内容替换为以下代码（请将 `APPGROUP` 替换为您自己的 Identifier）：

```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt

// 替换为您自己的 App Group Identifier
let APPGROUP = "group.com.tencent.comm.tcic.sharescreen"

class SampleHandler: RPBroadcastSampleHandler, TXReplayKitExtDelegate {

    let recordScreenKey = Notification.Name.init("TRTCRecordScreenKey")

    override func broadcastStarted(withSetupInfo setupInfo: [String :
NSObject]?) {
        TXReplayKitExt.sharedInstance().setup(withAppGroup: APPGROUP,
delegate: self)
    }

    override func broadcastPaused() {
        // 用户已请求暂停直播
    }
```

```
    override func broadcastResumed() {
        // 用户已请求恢复直播
    }

    override func broadcastFinished() {
        TXReplayKitExt.sharedInstance().finishBroadcast()
    }

    func broadcastFinished(_ broadcast: TXReplayKitExt, reason:
TXReplayKitExtReason) {
        var tip = ""
        switch reason {
        case TXReplayKitExtReason.requestedByMain:
            tip = "屏幕共享已结束"
        case TXReplayKitExtReason.disconnected:
            tip = "应用断开"
        case TXReplayKitExtReason.versionMismatch:
            tip = "集成错误（SDK 版本号不相符合）"
        default:
            break
        }

        let error = NSError(domain:
NSStringFromClass(self.classForCoder), code: 0, userInfo:
[NSLocalizedFailureReasonErrorKey:tip])
        finishBroadcastWithError(error)
    }

    override func processSampleBuffer(_ sampleBuffer: CMSampleBuffer,
with sampleBufferType: RPSampleBufferType) {
        switch sampleBufferType {
        case RPSampleBufferType.video:

TXReplayKitExt.sharedInstance().sendVideoSampleBuffer(sampleBuffer)
        case RPSampleBufferType.audioApp:
            // 处理应用程序音频
            break
        case RPSampleBufferType.audioMic:
            // 处理麦克风音频
            break
```

```
            @unknown default:
                fatalError("未知类型")
            }
        }
    }
```

参数说明

TCICView 组件参数

参数名	类型	是否必填	描述
`controller`	TCICController	是	控制器实例。
`config`	TCICConfig	是	核心配置信息。
`callback`	TCICCallback	否	课堂事件回调函数。

TCICConfig 核心配置

字段名	类型	是否必填	描述
`userId`	String	是	用户 ID。
`classId`	String	是	课堂 ID。
`token`	String	是	认证令牌。
`role`	RoleEnum	否	用户角色（学生/老师等）。
`langConfig`	TCICLangConfig	否	语言配置。
`nameConfig`	TCICNameConfig	否	角色名称自定义配置。
`fontConfig`	TCICFontConfig	否	字体配置。
`liveplayerConfig`	TCICLivePlayerConfig	否	直播播放器配置。
`isLatestBackend`	bool	否	是否使用最新后端。
`isTestBackend`	bool	否	是否使用测试后端。
`componentConfig`	List<TCICComponentConfig>	否	UI 组件自定义配置列表。

UI 组件配置 (TCICComponentConfig)

`TCICComponentConfig` 是一个抽象类，用于配置课堂界面中的各个 UI 组件。

使用示例：

```
final config = TCICConfig(
  userId: 'user123',
  classId: 'class456',
  role: RoleEnum.student,
  token: 'your_token_here',
  componentConfig: [
    HeaderComponentConfig(
      isShow: true,
      enableHandsUp: true,
      iconConfig: HeaderIconConfig(
        micIcon: 'custom_mic_icon.png',
        cameraIcon: 'custom_camera_icon.png',
      ),
    ),
    MemebersComponentConfig(enableStageUpDownAction: true),
    MessageComponentConfig(),
    SetttingComponentConfig(),
    VideoComponentConfig(),
    WhiteboardComponentConfig(),
  ],
);
```

HeaderComponentConfig (头部组件)

字段名	类型	描述
`isShow`	bool	是否显示头部组件（默认 `true`）。
`enableHandsUp`	bool	是否显示举手（默认 `true`）。
`enableScreenShare`	bool	是否显示屏幕共享（默认 `true`）。
`enableMessage`	bool	是否显示消息（默认 `true`）。
`enableCourseware`	bool	是否显示课件（默认 `true`）。
`enableSetting`	bool	是否显示设置（默认 `true`）。

enableMemberList	bool	是否显示花名册（默认 true ）。
showClassStatus	bool	是否显示课程状态（默认 true ）。
showClassTime	bool	是否显示课程时间（默认 true ）。
showOnlineMemberCount	bool	是否显示在线成员数量（默认 true ）。
showQuitButton	bool	是否显示退出按钮（默认 true ）。
iconConfig	HeaderIconConfig	自定义头部组件图标配置。
headerBuilder	Widget Function()	自定义整个头部组件。

SettingComponentConfig (设置组件)

字段名	类型	描述
enableAudioSetting	bool	是否启用音频设置（默认 true ）。
enableVideoSetting	bool	是否启用视频设置（默认 true ）。
enableGeneralSetting	bool	是否启用通用设置（默认 true ）。
showMemberJoinExitInfo	bool	是否显示成员加入/退出提示（默认 true ）。

WhiteboardComponentConfig (白板组件)

字段名	类型	描述
enableCreateBoard	bool	是否启用创建白板（默认 true ）。
enableBoardList	bool	是否启用白板列表（默认 true ）。
enableSwitchPage	bool	是否允许 PPT 课件翻页（默认 true ）。

说明：
其他组件如 MessageComponentConfig 、VideoComponentConfig 均支持通过 Builder 函数进行高度自定义。

回调函数 (TCICCallback)

通过传入 TCICCallback ，您可以监听课堂内的各类核心事件：

回调方法	触发时机	备注
`onJoinedClassSuccess`	加入课堂成功时	-
`onJoinedClassFailed`	加入课堂失败时	-
`beforeExitedClass`	退出课堂前	返回 `false` 可拦截/取消退出操作。
`afterExitedClass`	退出课堂后	-
`onKickedOffClass`	被踢出课堂时	-
`onMemberJoinedClass`	其他成员加入课堂时	-
`onMemberLeaveClass`	其他成员离开课堂时	-
`onReceivedMessage`	收到聊天消息时	-
`beforeRenderMessage`	渲染消息前	返回 `false` 可拦截该消息的 UI 渲染。

常见问题

1. 权限问题

- **Android 权限被拒绝：**
  - 检查 `AndroidManifest.xml` 中是否已添加相应权限。
  - 确保在代码中动态申请了运行时权限。
- **iOS 权限被拒绝：**
  - 检查 `Info.plist` 中是否已添加权限描述文本。
  - 确保权限描述文本清晰明了，否则可能被 App Store 拒绝。

2. 音视频问题

- **无法听到声音：**

  检查设备音量设置及麦克风权限是否已授权。
- **无法看到视频：**

  检查相机权限是否已授权，确认设备相机硬件功能正常。

3. 网络与连接问题

- **连接失败：**
  - 验证 `SDKAppId` 和 `token` 是否正确且未过期。
  - 检查设备网络连接及防火墙设置。
- **音视频卡顿：**

检查网络带宽是否充足，考虑在设置中降低音视频质量。

4. 白板问题

- **白板无法加载：**

  检查网络连接状态，验证白板配置参数是否合法。
- **白板操作无响应：**

  确认当前用户的角色权限（如观众角色默认无白板操作权限）。

uni-app 快速接入

最近更新时间：2026-06-08 18:02:30

本文档将指导您如何在 uni-app 项目中快速接入腾讯云实时互动-教育版（LCIC）SDK。

开发环境要求

在开始之前，请确保您的开发环境满足以下要求，并建议您提前阅读 uni-app 原生语言插件使用教程，了解如何引入原生插件。

- **iOS**：系统版本要求 11.0 及以上。
- **Android**：API Level 要求 21 及以上。

Demo 及源码体验

为了帮助您快速了解并体验集成了腾讯云实时互动-教育版的 uni-app 应用，我们提供了完整的 Demo 及配套源码。

- **获取源码**：您可以访问 GitHub 仓库 获取完整源码。
- **快速体验 Demo**：如果您使用的是 Android 设备，可直接扫描下方二维码，下载并安装我们打包好的 uni-app 体验包。

体验说明：

进入应用后（如下图所示），您可以直接复制 线上 Demo 创建的课堂 链接进入课堂。

> **说明：**
> 您也可以手动在链接末尾拼接 Token 参数（格式为 `&token=$TOKEN`，Token 可通过按 `F12` 打开控制台，在 Network 的课堂请求 Payload 中获取），将其贴入输入框，系统将自动解析参数。

接入步骤

步骤1：获取必要信息

请参考 接入准备 创建应用并获取必要信息。

> **说明：**
> 移动端接入需要购买**旗舰版**或**企业尊享版**，请创建对应版本的应用。

步骤2：导入 SDK

1. **引入 TRTC 插件**：前往插件市场购买（免费）腾讯云实时音视频 SDK（插件 1），并绑定至您的项目。
2. **引入 LCIC 插件**：前往 GitHub 仓库 下载**腾讯云实时互动-教育版 uni-app 插件包**（插件 2）。将压缩包内的 `lcic-sdk-pro` 文件夹放入您本地项目的 `nativeplugins` 目录下（若无此目录请手动创建）。
3. **配置 manifest**：在 HBuilderX 中打开项目，进入 `manifest.json` 的 **App 原生插件配置**，勾选上述两个插件。

> **说明：**
> **更新建议：**Native SDK 会不定期迭代，建议您持续关注 GitHub 仓库 的更新动态，及时下载最新包并替换本地 `nativeplugins` 中的旧版本。

步骤3：SDK 授权申请

App 开发需要获取以下两种 License：

1. Web 引擎 License（Android 平台需要）

- **测试环境：**请参考 申请 Web 引擎测试用 License，**完成流程至第四部分获取 LicenseKey 即可。**
- **正式环境：**购买旗舰版后，请通过 联系我们 加入微信群联系技术支持申请正式版 License。

2. 播放器授权（仅“直播大班课”需要）

如果是“直播大班课”班型，需申请播放器权限。请按以下模板填写信息并提交 腾讯云工单，相关疑问可通过 联系我们 加入微信群了解。

> **注意：**
> 一个旗舰版仅支持授权一个正式包名，请确认信息无误。

参考模板提供信息：

- 问题：实时互动-教育版申请播放器授权
- 公司名称：
- 联系方式：
- App Name：
- Package Name（Android）：
- Bundle ID（iOS）：

步骤4：初始化 Web 引擎

相比于 Android 系统自带的 WebView，Web 引擎在兼容性和加载速度上具有显著优势。

1. **合规性检查**：请务必在用户**同意隐私政策协议之后**，再调用 Web 引擎初始化方法，以避免应用上架时因“未经同意收集个人信息”被拒。
2. **执行初始化**：在进入课堂前，**必须保证该方法执行完毕**。若初始化成功，将使用 Web 引擎进入课堂；若初始化失败，SDK 会自动降级采用系统 WebView 兜底方案，不影响正常进入课堂。

```
const lcicModule = uni.requireNativePlugin('lcic-sdk-pro');

lcicModule.initX5Core({
    licenseKey: "您的 Web 引擎 LicenseKey" // 见步骤3获取的 LicenseKey
}, (ret) => {
    if (ret && ret.code === 0) {
        console.log("Web 引擎初始化成功，后续将采用 Web 引擎方案");
    } else {
        console.warn("Web 引擎初始化失败，后续将采用系统 WebView 兜底方案");
    }
});
```

> **说明：**
> 若持续出现 Web 引擎初始化失败的情况，可通过 联系我们 加入微信群联系技术支持排查。

步骤5：获取进入课堂所需参数

调用进入课堂接口前，需准备相关参数。参数来源如下：

字段名	类型	必填	含义	获取方式 / 备注
`schoolId`	int	是	学校编号	即控制台中的 SDKAppId 。
`classId`	long	是	课堂编号	通过服务端 API CreateRoom 创建课堂后返回的 `RoomId` 。
`userId`	string	是	用户账号	通过服务端 API RegisterUser 注册获取。
`token`	string	是	鉴权 Token	通过服务端 API LoginUser 登录获取。
`scene`	string	否	场景名称	用于区分不同的定制布局，有两种配置方式： 方式1：使用 SetAppCustomContent 接口配置。 方式2：控制台 场景配置 。

`lng`	string	否	语言参数	界面语言设置。支持值： • `zh-CN`：中文简体 • `zh-TW`：中文繁体 • `en-US`：英语 • `ja`：日语 • `ko`：韩语 • `ar`：阿拉伯语 • `vi`：越南语 • `id`：印尼语
`camera`	int	否	初始摄像头状态	`1`：开启（默认），`0`：关闭。
`mic`	int	否	初始麦克风状态	`1`：开启（默认），`0`：关闭。
`speaker`	int	否	初始扬声器状态	`1`：开启（默认），`0`：关闭。

步骤6：调起组件主页面

传入上述获取的参数，即可唤起 LCIC 课堂主页面。

```
const lcicModule = uni.requireNativePlugin('lcic-sdk-pro');

lcicModule.joinClass({
    schoolId: schoolId, // 等同于 SDKAppId
    classId: classId,
    userId: userId,
    token: token,
    // 可选参数
    // camera: 1,
    // mic: 1
}, (res) => {
    // 进入课堂的回调处理
    console.log("进入课堂结果：", res);
});
```

步骤7：编译与运行

- **调试阶段**：建议您制作包含 TRTC 和 LCIC 两个插件的**自定义调试基座**进行真机运行调试。

- **发布阶段**：调试无误后，可使用 HBuilderX 的云打包功能，打包导出最终的 App 安装包。

定制化开发
定制化概述

最近更新时间：2026-03-18 16:41:02

为了满足不同教育机构和在线课堂场景的个性化需求，我们为您提供了高度灵活的线上实时音视频上课 SDK 解决方案。通过我们的定制化能力，您可以轻松打造符合自身品牌调性、业务逻辑和用户习惯的专属在线教室。
本文档旨在为您提供移动端 SDK 定制化能力的全局概览，并引导您快速找到所需的开发资源。

移动端 SDK 架构：Web 内核 vs 原生内核

为了适应不同的开发团队技术栈以及业务对性能、灵活性的不同侧重，我们的移动端 SDK 提供了两种底层架构供您选择：**Web 内核**与**原生内核**。
无论您选择哪一种内核架构，我们都为其配备了丰富的定制化能力，涵盖了从基础的 UI 界面修改到深度的音视频交互逻辑调整。
如果您在技术选型阶段，需要详细了解这两种内核在性能表现、开发效率、跨平台能力等方面的差异，请参见 移动端原生内核与 Web 内核的区别 。

Web 内核定制化

Web 内核 SDK 依托于成熟的 Web 技术栈，具有极高的跨平台灵活性和动态更新能力。它非常适合拥有前端开发经验的团队，能够以极低的成本实现界面的快速迭代。
定制化能力亮点：

- **UI/UX 灵活调整：**支持通过 CSS/HTML 快速修改课堂主题色、按钮样式、布局排版等。
- **动态下发：**业务逻辑和界面的修改可以实现热更新，无需用户频繁升级 App。
- **丰富的 JS API：**提供完善的 JavaScript 接口，方便与您现有的 Web 业务系统进行无缝对接。

开发指南：
如果您选择使用 Web 内核，并希望了解具体的定制化接入步骤、API 列表及代码示例，请参见 Web 内核定制指南 。

原生内核定制化

原生内核 SDK 深度结合了 iOS 和 Android 系统的底层特性，能够最大程度地压榨硬件性能，提供出色的音视频低延迟体验和丝滑的交互感受。它适合对课堂性能、弱网抗性以及底层硬件控制有极高要求的业务场景。
定制化能力亮点：

- **出色性能体验：**深度优化的原生音视频引擎，保障百人、千人大型互动课堂的稳定运行。
- **原生 UI 组件：**提供高度可定制的原生 UI 控件，支持通过 Swift/Objective-C、Kotlin/Java 或 Flutter 进行深度重写。

开发指南：
如果您选择使用原生内核，并希望了解如何在 iOS 或 Android 平台上进行深度定制开发，请参见 原生内核定制指南 。

技术支持

在定制化开发的过程中，如果您遇到任何技术难题、API 使用疑问，欢迎通过 联系我们 加入官方开发者微信群。我们的技术与产品专家将在线为您答疑解惑。

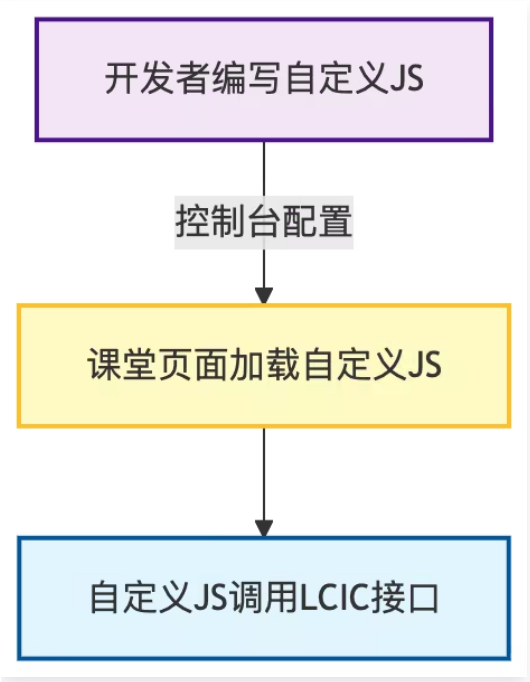
Web 内核定制指南
快速开始定制（Web 内核）

最近更新时间：2026-03-27 14:28:22

参考本文档前请确保您已经了解并完成了 接入准备 和 Web 及 H5 快速接入 的基础集成。

核心原理：定制化开发的运作逻辑

互动课堂的各端界面（Web、Electron、Web 内核的 iOS/Android），**底层都是基于网页（Web）实现的**。
定制化开发的**核心逻辑**是：
不修改 SDK 内核源码，通过注入自定义 JS/CSS 文件，借助 SDK 开放的全局对象 TCIC.SDK.instance，在课堂生命周期的对应节点，实现界面样式修改、业务逻辑扩展、功能权限管控，最终实现全端统一的自定义效果。



定制化原理

通过定制化能实现什么？

通过注入自定义的 JS 和 CSS，您可以**修改所有端的界面并补充业务逻辑**。常见场景如下：

业务方向	具体场景举例
界面与样式	**文案替换**：将页面上的“课堂”字样全部替换为“会议”等行业专有词。 **布局调整**：修改摄像头/麦克风图标样式、调整弹窗按钮行为、自定义虚拟背景等。
功能与权限	**屏蔽功能**：隐藏头像区域、隐藏在线人数等无关业务。 **权限配置**：针对不同角色（老师/学生）配置白板权限、设置下课倒计时提醒等。

数据与事件	**事件监听**：当特定事件发生时（如进入房间、开关麦克风），触发业务弹窗，或将状态上报给您的业务后台。

快速开始：本地开发与调试

无论您是使用官方模板还是从零开发，调试的核心步骤都是**启动本地服务 > 拼接课堂 URL > 注入代码**。

步骤1：准备本地 JS/CSS 文件

您可以选择以下两种方式之一来准备代码：

方式一：使用官方 Demo

我们提供了常用的自定义示例代码，开箱即用：

```
git clone https://github.com/InteractiveClassroom/tcic-custom-demo.git
pnpm i
cd ./demos/{SOME_DEMO} # 选择一个您需要的 demo 目录
pnpm dev # 默认会在 localhost:3000 启动服务
```

方式二：自己从零搭建

如果您熟悉前端开发，可以直接在本地搭建一个静态服务器（假设运行在 `8080` 端口），并在根目录创建 `test.js` 和 `test.css`。

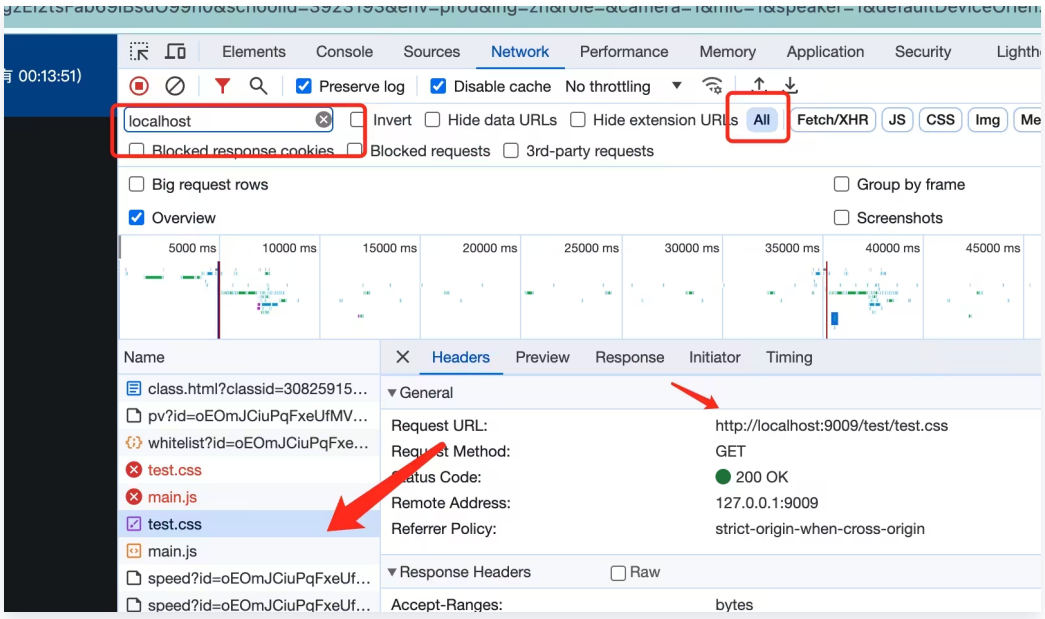
> **注意：**
> 如果使用 Vue/React 开发，请自行创建 div 节点插入 DOM，不要直接挂载到课堂原有的 `#app` 节点上。

步骤2：在课堂中注入并调试

1. 进入 课堂演示登录页，依次单击**创建课堂** > **立即进入**。
2. 进入课堂后，复制浏览器地址栏中的 URL。
3. 在 URL 末尾拼接您的本地文件地址：
   - 拼接 JS：`&debugjs=http://localhost:3000/custom.js`
   - 拼接 CSS：`&debugcss=http://localhost:3000/custom.css`

- 完整示例：`https://class.qcloudclass.com/...&debugjs=http://localhost:3000/custom.js&debugcss=http://localhost:3000/custom.css`

4. 敲击回车键重新加载页面。打开开发者工具（F12）的 Network 面板，如果看到您的本地 JS/CSS 文件被成功加载，说明环境准备完毕，可以开始写代码了。



步骤3：发布到生产环境

当您在本地（`localhost`）调试完毕，确认功能无误后，就可以发布上线了：

1. 在本地项目中运行打包命令（如 `npm run build`），生成最终的 JS 和 CSS 文件，并发布到您的生产环境，使外网能访问到。
2. 登录腾讯云控制台，进入自定义场景配置页面，详见 场景配置 。
3. 配置您打包好的 JS 和 CSS URL，并生成一个“场景名称”。
4. 以后在进入课堂的时候，传入这个“场景名称”，该课堂就会自动加载您的定制化代码。

Web/H5

参考 Web 及 H5 快速接入 在拼接进入课堂的时候，带上参数 `scene={场景名称}`，示例：

```
https://${课堂域名}/latest/class.html?
scene=${scene}&schoolid=${schoolid}&classid=${classid}&userid=${userid
}&token=${token}
```

安卓（Web 内核）

参考文档 Web 内核（Android）快速接入 在进入课堂时，传参 `scene`，示例：

```
// 1. 构建配置对象
TCICClassConfig initConfig = new TCICClassConfig.Builder()
    .schoolId(schoolId)  // 应用 ID
    .classId(classId)    // 课堂 ID
    .userId(userId)      // 用户 ID
    .token(token)        // 鉴权 Token
    .scene(scene)        // 场景名称
    .build();
```

iOS（Web 内核）

参考 Web 内核（iOS）快速接入 在进入课堂时，传参 `scene`，示例：

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];

// 必填参数
roomConfig.schoolId = 123456;        // 学校/应用 ID
roomConfig.userId = "test_user";    // 用户 ID
roomConfig.token = "test_token";    // 鉴权 Token
roomConfig.classId = 654321;         // 课堂 ID

// 可选参数配置
[roomConfig setValue:@"en" forKey:@"language"];          // 设置语言
[roomConfig setValue:@"scene_name" forKey:@"scene"];     // 设置场景
```

Electron

参考 Electron 快速接入 在进入课堂的时候，带上参数 `scene={场景名称}` 。
示例一：
URL 唤起方式，需要拼接 URL。

```
tcic://${课堂域名}/latest/class.html?
scene=${scene}&classid=${classId}&userid=${userId}&token=${token}
```

示例二：
SDK 集成方式，在初始化时传入 `scene` 。

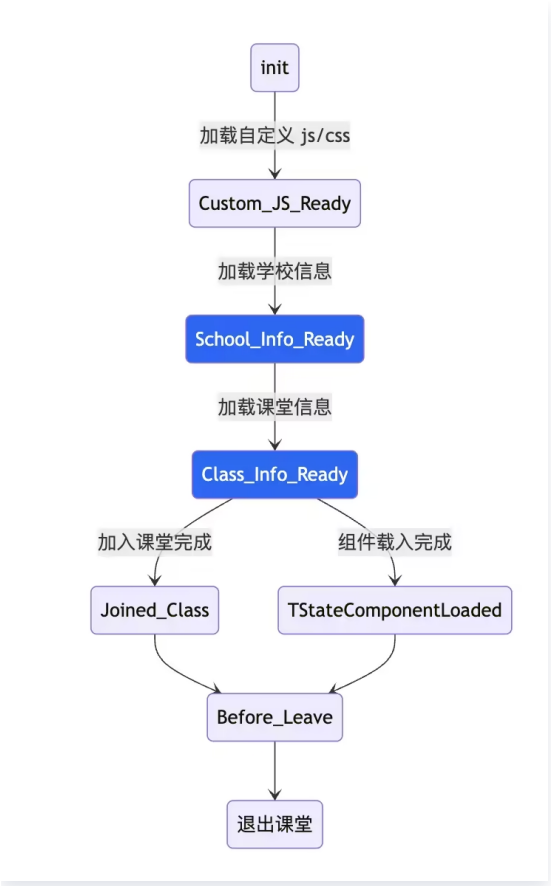
```
const TCIC = require('tcic-electron-sdk');

TCIC.initialize({
  // 核心鉴权参数
  classId: '368507569',
  userId: '123456',
  token: 'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...', // 务必动态获取

  // 可选：自定义参数（透传至 Web 端）
  customParams: {
    key: 'value',
    scene: 'scene_name',  //自定义场景名称
  },
  ...
})
```

基本概念与 API

在您的自定义 JS 文件中，主要通过操作**状态（State）**、**事件（Event）**和**功能开关（Feature）**来实现业务逻辑。

1. 课堂生命周期与状态

课堂的加载是一个按顺序进行的过程。**绝大多数自定义逻辑，都必须等课堂加载到特定状态后才能执行**，否则会报错。



课堂生命周期

关键状态节点：

含义	状态名	类型	备注
学校信息已获取	TCIC.TMainState.School_Info_Ready	Boolean	自定义 js/css 地址是从获取学校信息接口中获取，js/css 加载完成后，School_Info_Ready 状态会变成 true。
课堂信息已获取	TCIC.TMainState.Class_Info_Ready	Boolean	获取到课堂信息后，会处理布局和文案相关内容。 **注意：**除了 替换关键文案 等少量自定义需求需要在 Class_Info_Ready 前执行外，自定义代码的逻辑都应该在 Class_Info_Ready 状态之后再执行。
加入 TRTC 房间	TCIC.TMainState.Joined_TRTC	Boolean	在该状态改变后，用户才能使用 TRTC 房间相关功能，例如开启麦克风或者打开视频。
是否进行设备检测	TCICUI.Constant.TStateDeviceDetect	Boolean	需要连麦视频的用户完成设备检测再进入房间可以减少连麦异常的情况。

说明：

> 历史原因状态值有两类，挂载在 TCIC.TMainState 和 TCICUI.Constant 对象，具体参考 TCIC-SDK API 文档。

如何使用状态代码：

我们强烈推荐使用 `promiseState` 方法，它能确保当状态满足条件时，安全地执行您的代码。

```
// 设置状态值
TCIC.SDK.instance.setState(name ,val)
// 示例：promiseState可以确保当前状态满足条件的时候立即执行一次。
TCIC.SDK.instance.promiseState(name,val)
// 示例：获取状态值
TCIC.SDK.instance.getState(name)

// 示例：当“学校信息已获取”时，打印学校信息
TCIC.SDK.instance
  .promiseState(TCIC.TMainState.School_Info_Ready, true)
  .then(() => {
    console.log("学校信息:", TCIC.SDK.instance.getSchoolInfo());
  });

// 示例：强制关闭进入课堂前的“设备检测”环节
TCIC.SDK.instance.setState(TCICUI.Constant.TStateDeviceDetect, false);
```

2. 课堂事件监听

除了状态，课堂还会不断抛出各种事件（例如有人进房、收到消息等）。您可以通过 `on` 和 `off` 来监听或取消监听。

常用事件举例：

- `TCIC.TMainEvent.Show_Msg_Box`：弹窗出现时触发，使用示例可以参考 修改下课按钮行为。
- `TCIC.TMainEvent.Member_Join`：有人加入课堂时触发。
- `TCIC.TIMEvent.Recv_Msg`：收到聊天消息时触发。

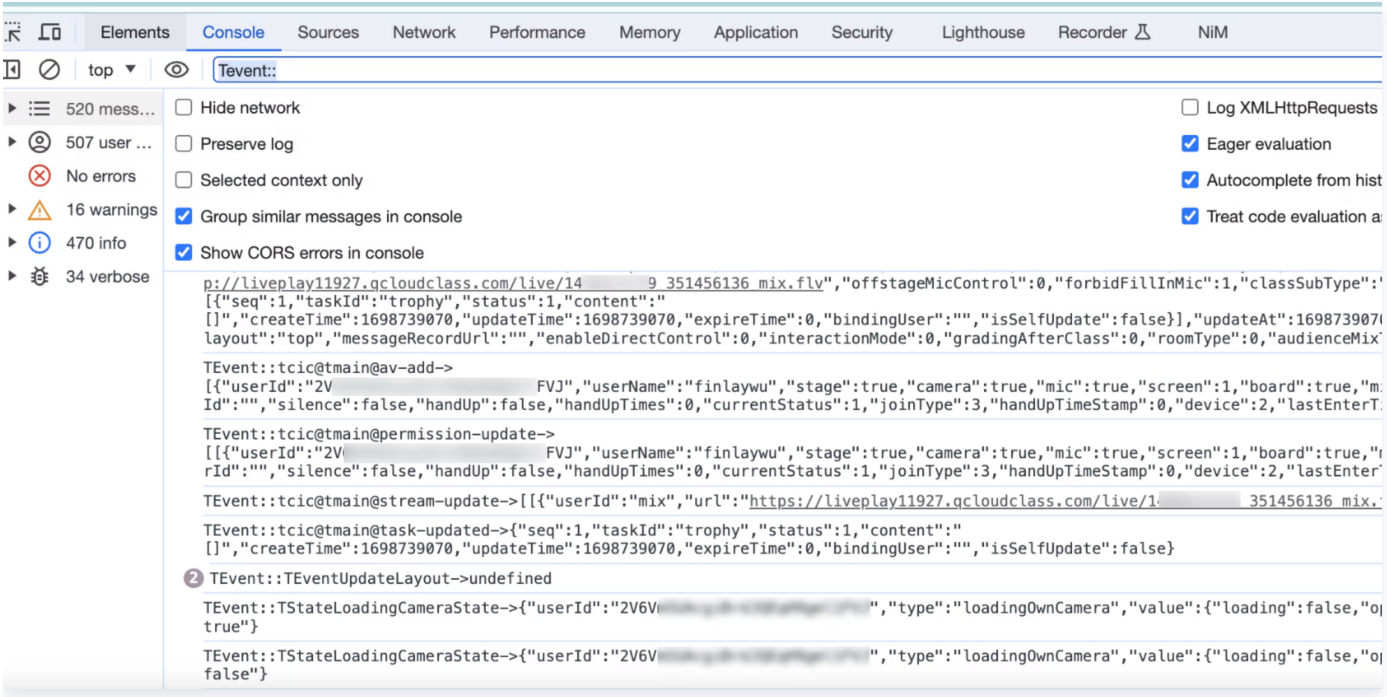
事件可以在 TCIC.TMainEvent/TCIC.TIMEvent 里找到。

```
// 示例：监听事件
TCIC.SDK.instance.on(eventname, callback);
// 示例：取消监听事件
TCIC.SDK.instance.off(eventname, callback);

// 示例：监听收到消息事件
TCIC.SDK.instance.on(TCIC.TIMEvent.Recv_Msg, (msg) => {
```

```
    console.log("收到新消息:", msg);
});
```

说明：

在课堂控制台过滤 `Tevent::` 可以观察到所有实时触发的事件行为。



3. 功能开关配置

针对不同的角色，您可能需要开启或关闭某些特定功能。这可以通过 `setFeatureAvailable` 来实现。此操作应在课堂信息加载完成（`Class_Status`）之后进行。

常用白板开关：

- `WhiteBoardList`：是否允许新增/切换白板（默认 true）。
- `WhiteBoardPPT`：是否允许课件翻页（默认 true）。
- `WhiteBoardPPT.WheelPaging`：是否支持用户使用鼠标滚轮对课件翻页（默认 false）。

```
// 示例：如果是学生，则限制其白板操作权限
TCIC.SDK.instance
  .promiseState(TCIC.TMainState.Class_Status,
TCIC.TClassStatus.Already_Start)
  .then(() => {
    if (TCIC.SDK.instance.isStudent()) {
      TCIC.SDK.instance.setFeatureAvailable("WhiteBoardList", true); // 允
许添加白板
```

```
        TCIC.SDK.instance.setFeatureAvailable("WhiteBoardPPT", false); // 禁
止翻页
        TCIC.SDK.instance.setFeatureAvailable("WhiteBoardPPT.WheelPaging",
false); // 禁止滚轮翻页
    }
  });
```

更多参考资料

本文仅介绍了最核心的定制化原理和常用 API。如需查阅完整的状态枚举、事件列表和方法详情，请参见 TCIC-SDK API 官方文档 。

事件监听（Web 内核）

最近更新时间：2026-03-18 16:41:01

使用场景

在实际业务需求中，您可能需要在特定事件发生时进行与业务相关的处理，例如：

- 当上课正式开始时，向业务后台进行一些上报处理。
- 当成员加入房间时，向成员展示弹窗。

您可以通过 `TCIC.SDK.instance.on(eventName, handler)` 监听成员进入退出等课堂事件，或者通过 `TCIC.SDK.instance.subscribeState(stateName, handler)` 监听课堂状态变更。

事件列表（TCIC.TMainEvent）

Event	事件
After_Enter	已加入房间。
Modify_Class	房间信息已更改。
Leave_Class	离开房间。
Kick_Out_By_Teacher	被踢出房间。
Kick_Out_By_Another	多端登录被踢出房间。
Kick_Out_By_Expire	签名过期被踢出房间。
Member_Join	成员加入房间。
Member_Exit	成员退出房间。
Member_Info_Update	成员信息更新。
Member_Hand_Up	成员举手。
Member_Hand_Up_Cancel	成员取消举手。
Question_Valid	存在可用答题。
Question_Begin	答题开始。
Question_End	答题结束。
Question_Abandon	终止答题。
Question_Close	关闭答题。

Question_Been_Answered	有学生作答。
App_Resized	应用大小变化。
Error	发生错误(影响主线流程)。
Recv_IM_Msgs	收到 IM 消息。
Recv_Custom_IM_Msg	收到自定义 IM 消息。

使用示例：

```
function afterEnter() => {
  console.debug('You have joined this room');
}

// 监听
TCIC.SDK.instance.on(TCIC.TMainEvent.After_Enter, afterEnter);

// 取消监听
TCIC.SDK.instance.off(TCIC.TMainEvent.After_Enter, afterEnter);
```

状态列表（TCIC.TMainState）

Event	事件	说明
Class_Info_Ready	课堂信息已加载。	–
Joined_Class	已加入课堂。	–
Sub_Camera	辅助摄像头状态。	• 0：开始 • 2：结束
Screen_Share	屏幕分享状态。	• 0：分享中 • 1：暂停中 • 2：未开始/已结束
Video_Publish	本地视频推流是否开启。	–
Audio_Capture	本地音频采集是否开启。	–
Class_Duration	课堂持续时间。	单位秒。 • < 0：未到上课时间，距离上课开始的时间。

		• = 0：到上课时间未开始上课、课堂已结束、课堂已过期。 • > 0：上课中，已开始上课的时间。
Member_Count	课堂当前在线成员数量。	仅学生在线数，不包含老师、助教和巡课。
Member_List_Total_Member_Count	成员总数。	包括老师，助教在内，在线和离线的成员总数。
Member_List_Offline_Member_Count	离线成员总数。	包括老师，助教在内，离线的成员总数。
Board_Permission	白板操作权限。	-
Chat_Permission	文字聊天权限。	-
Screen_Share_Permission	屏幕分享权限。	-
Hand_Up	举手状态。	-
Mute_All	全员静音状态。	-
Mute_Video_All	全员视频状态。	-
Silence_All	全员禁言状态。	-
Message_Unread_Count	未读消息。	-
HandUp_Count	举手人数。	-
Class_Status	课堂状态。	• 0：未开始 • 1：已开始 • 2：已结束 • 3：已过期

```
// promiseState 可以确保当前状态满足条件的时候立即执行一次
TCIC.SDK.instance.promiseState(TCIC.TMainState.Joined_Class, true).then(
() => {
  console.debug('You have joined this room');
});


function listener() {
```

```
    console.debug('You have joined this room');
}
// 监听
TCIC.SDK.instance.subscribeState(TCIC.TMainState.Joined_Class, listener);

// 取消监听
TCIC.SDK.instance.unsubscribeState(TCIC.TMainState.Joined_Class,
listener);
```

定制化示例
替换关键文案

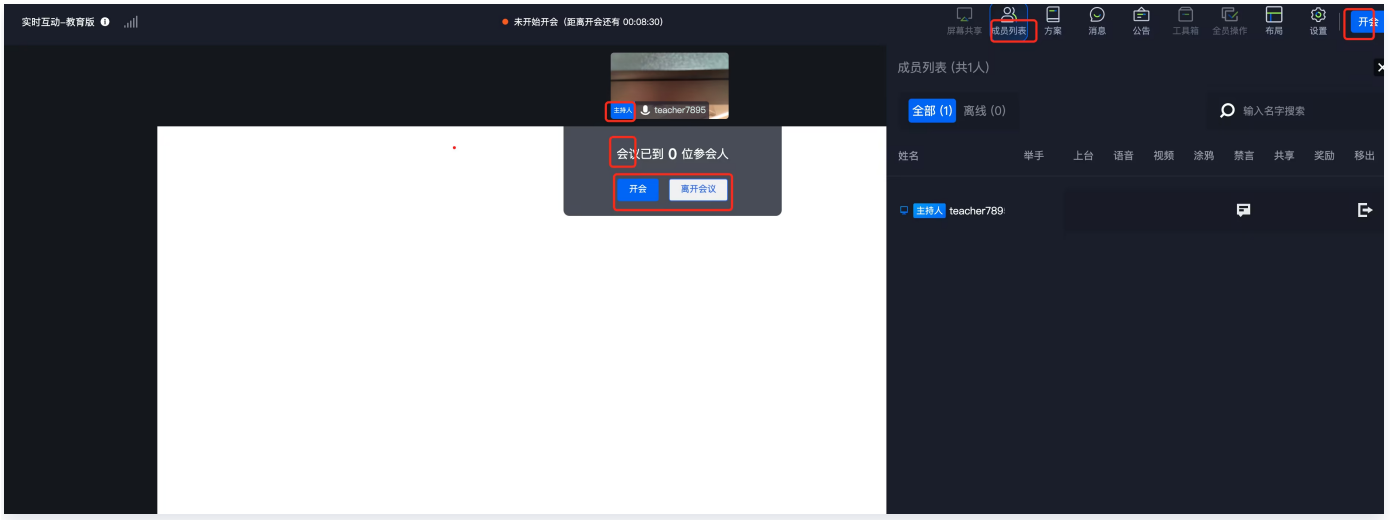
最近更新时间：2026-03-18 16:41:01

> **说明：**
> 阅读本章节前，请确保您已经了解 快速开始定制 的内容。

功能场景

更新课堂/直播间的专有名词，以更好适配客户自己的业务场景。

页面位置



该产品主要适用于教育环境，因此大多数文本内容与课程有关。但我们也提供文本替换功能以满足更多需求。

操作方法

在 快速开始定制 创建的 `test.js` 文件中，粘贴以下代码。刷新页面即可查看效果。
例如，更换会议场景中用到的专有名词。

```
/**
 * 注入JS会在全局环境下找到TCIC对象
 * TCIC.SDK.instance 为SDK实例
 * TCIC.TMainState 为SDK状态枚举
 * School_Info_Ready 为学校信息加载完成
 * promiseState 表示当状态为School_Info_Ready时，必定会执行一次
 *
 * 注意一定要在School_Info_Ready状态下才能调用initNameConfig
```

```
 */
TCIC.SDK.instance
  .promiseState(TCIC.TMainState.School_Info_Ready, true)
  .then(() => {
    console.log("[customJS] initNameConfig");
    /**
     * 配置关键文案
     */
    TCIC.SDK.instance.initNameConfig({
      zh: {
        teacher: "主持人",
        assistant: "联席主持人",
        supervisor: "巡会",
        student: "参会人",
        visitor: "访客",
        defaultAppName: "低代码互动会议",
        room: "会议",
        publicRoom: "公开会议",
        roomID: "会议号",
        roomName: "会议名",
        startRoom: "开会",
        endRoom: "结束会议",
        enterRoom: "进入会议",
        leaveRoom: "离开会议",
        courseware: "方案",
        memberList: "成员列表",
      },
    });
  });
```

其它文档

更多 SDK 方法请参考 TCIC-SDK API 文档 。

修改下课按钮行为

最近更新时间：2026-03-18 16:41:01

当我们希望老师不能提前下课，那么就可以修改下课按钮的行为。

> **说明：**
> 阅读本章节前，请确保您已经了解 快速开始定制 的内容。

默认弹窗

课堂默认展示的下课弹窗如下图：

操作方法

粘贴以下代码到 快速开始定制 里准备好的 `test.js` 中，即可实现修改下课按钮。

```
TCICCustomUI.hooks(TCICCustomUI.THookType.MsgBox_LeaveClass_BeforeShow).ta
p((event) => {
  if (TCIC.SDK.instance.isAssistant()) {
    event.payload.title = '温馨提示';
    // 修改弹窗内容
    event.payload.message = '助教，您确定要离开课堂吗？';
    // 不展示下课按钮
    event.payload.buttons.shift();
  }
});
```

其它文档

更多 SDK 方法请参考 TCIC-SDK API 文档 。

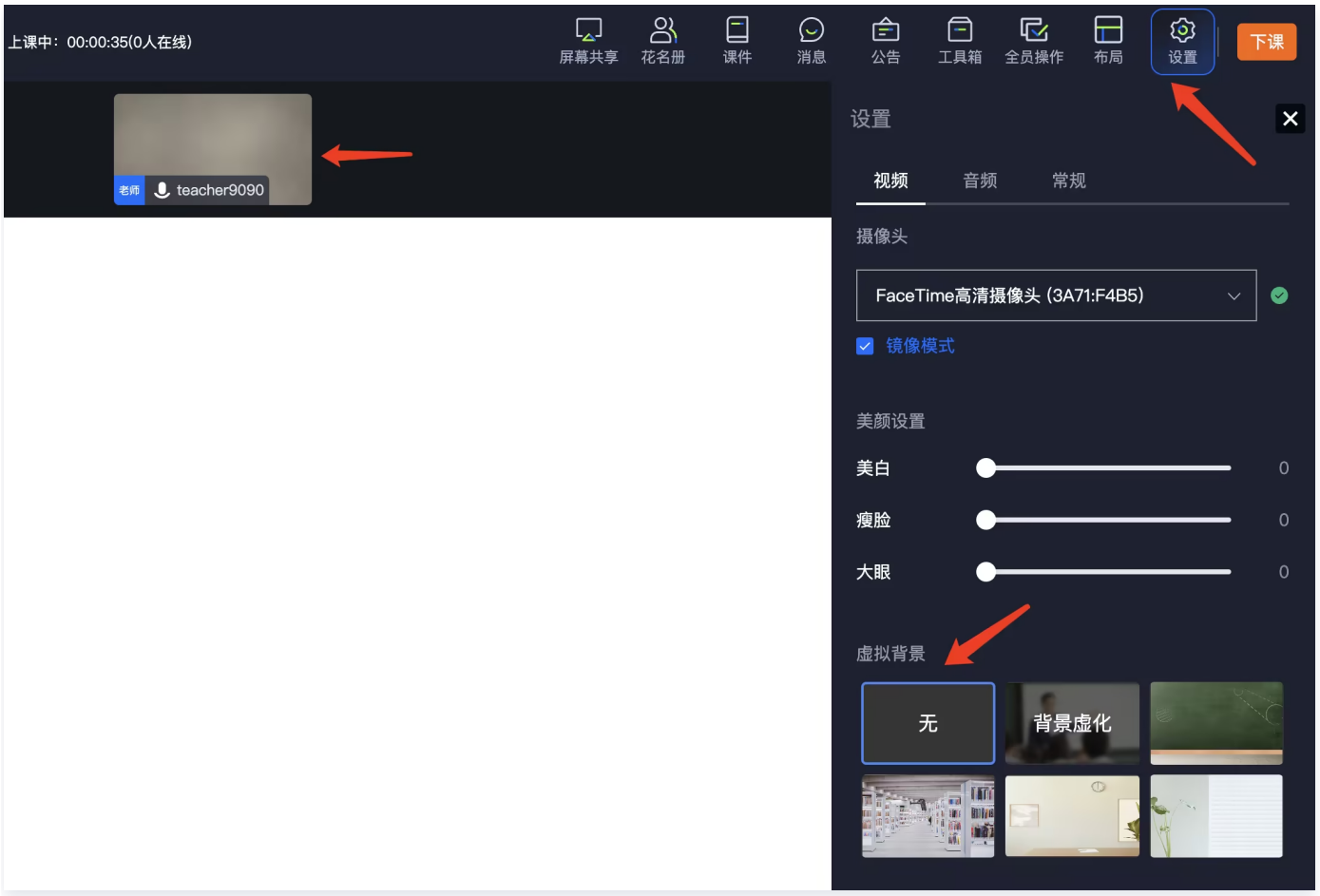
修改虚拟背景

最近更新时间：2026-03-18 16:41:01

> **说明：**
> 阅读本章节前，请确保您已经了解 快速开始定制 的内容。如果需要修改虚拟背景，参见下文说明。

通过设置面板修改

单击右上角**设置**，随后在出现的面板直接选择预设的虚拟背景即可生效。



使用自定义的虚拟背景

粘贴以下代码到 快速开始定制 里准备好的 `test.js` 中，通过 SDK 实例的方法进行设置，设置后会直接将设置的图片作为虚拟背景。

```
/**
 * 注入 JS 会在全局环境下找到 TCIC 对象
 * TCIC.SDK.instance 为 SDK 实例
 * @param enable  开关 true | false
```

```
 * @param url   虚拟背景图片地址，为空时为背景虚化
 */
TCIC.SDK.instance.setVirtualImg(enable, url);
```

添加其他自定义虚拟背景（支持用户自己按需选择）

粘贴以下代码到 快速开始定制 里准备好的 `test.js` 中，通过下方 SDK 实例方法进行设置，即可实现在预设的虚拟背景基础上，添加自有的虚拟背景，供用户自己选择。

```
TCIC.SDK.instance.promiseState(TCIC.TMainState.Video_Publish,
true).then(() => {
            setTimeout(() => {
                const dom = document.querySelector('.raw-virtual-
background-select-component');
                const vueInstance = dom.__vue__;
                // / 下面可以添加多个虚拟背景
                vueInstance.imgArr.push({
                    name: '背景',
                    url: 'https://static.sxqgrow.cn/upload/beijing1.jpg',
                    sceneKey: '背景',
                });
                // / 这里是默认选中哪一个虚拟背景
                vueInstance.onSelect(vueInstance.imgArr.length - 1);
            }, 300);
    });
```

> ⚠ **注意：**
> 请确保自定义背景图片的 URL 配置了正确的跨域头 **Access-Control-Allow-Origin**，以保证虚拟背景可以正常加载。

其它

更多 SDK 方法请参见 TCIC-SDK API 文档 。

屏蔽无关功能

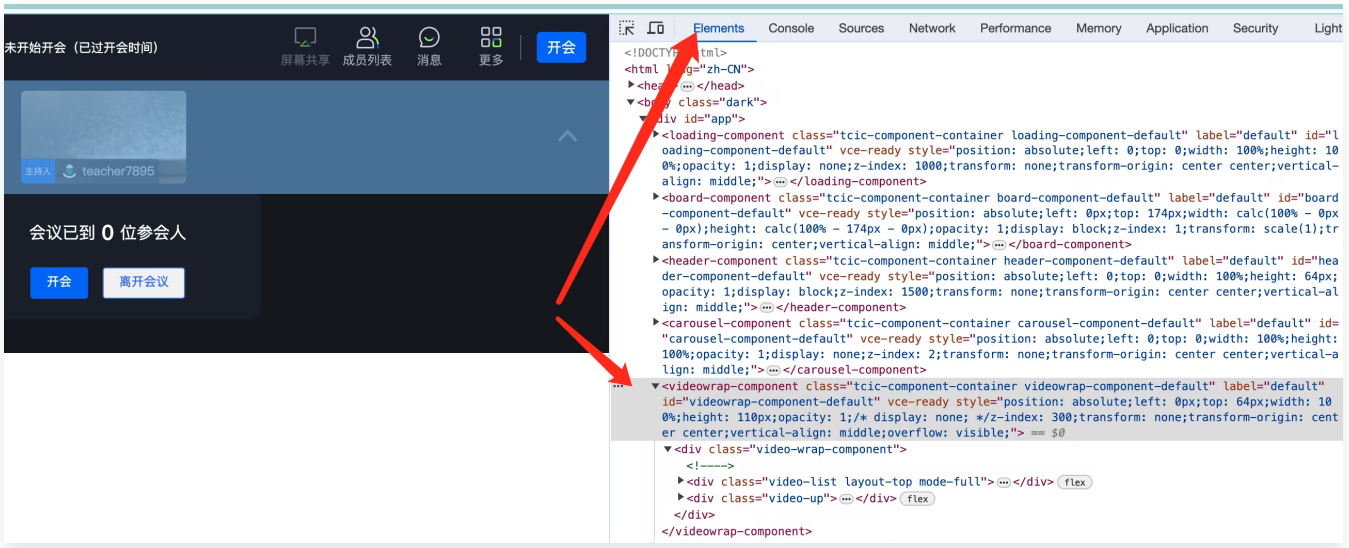
最近更新时间：2026-03-18 16:41:01

为了帮助用户更专注，我们会精简课程页面的内容，以实现更优质的产品体验。隐藏部分功能模块是一种有效的策略，本文档将展示两个隐藏功能的示例。

> **说明：**
> 阅读本文前，请确保您已经了解 快速开始定制 的内容。

示例一：CSS 文件屏蔽

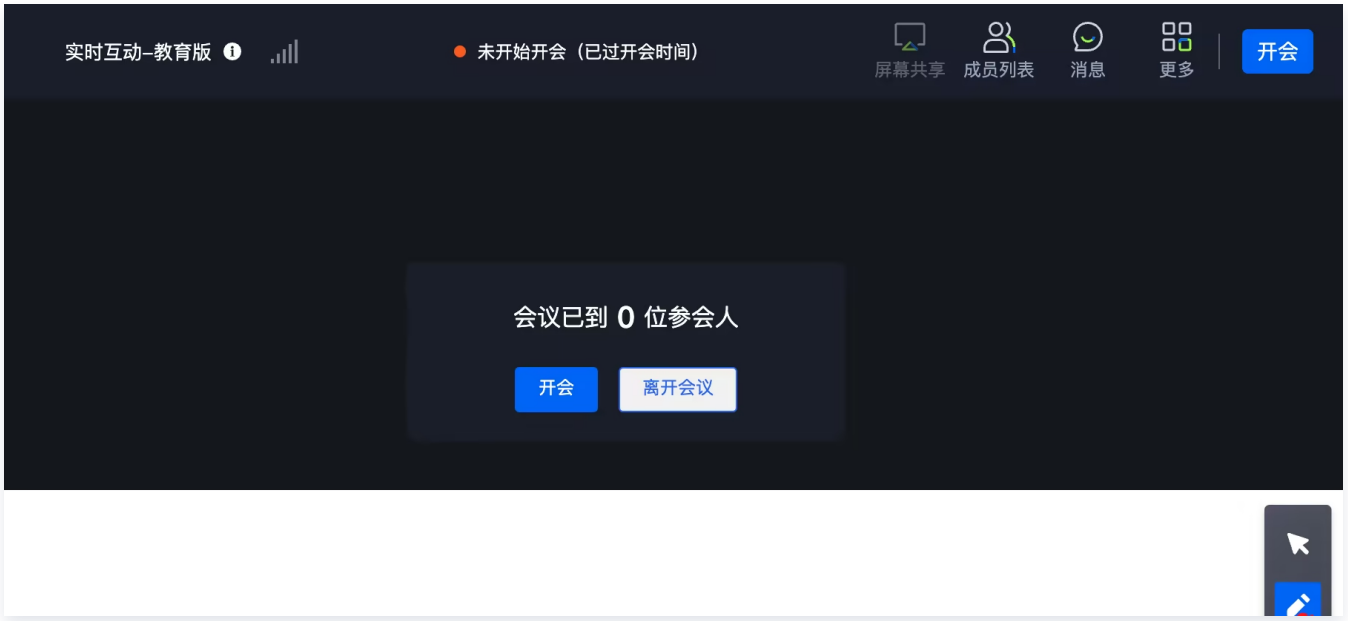
1. 首先打开浏览器控制台，鼠标右键查看元素，查找我们希望屏蔽的元素。以图中元素为例。



2. 观察到有 js 在元素上修改样式，所以我们用如下代码写入 快速开始定制 里准备好的 test.css。

```
.videowrap-component-default {
  /**使用 !important 覆盖指定元素上的样式规则 **/
  display: none !important;
}
```

修改效果如下：

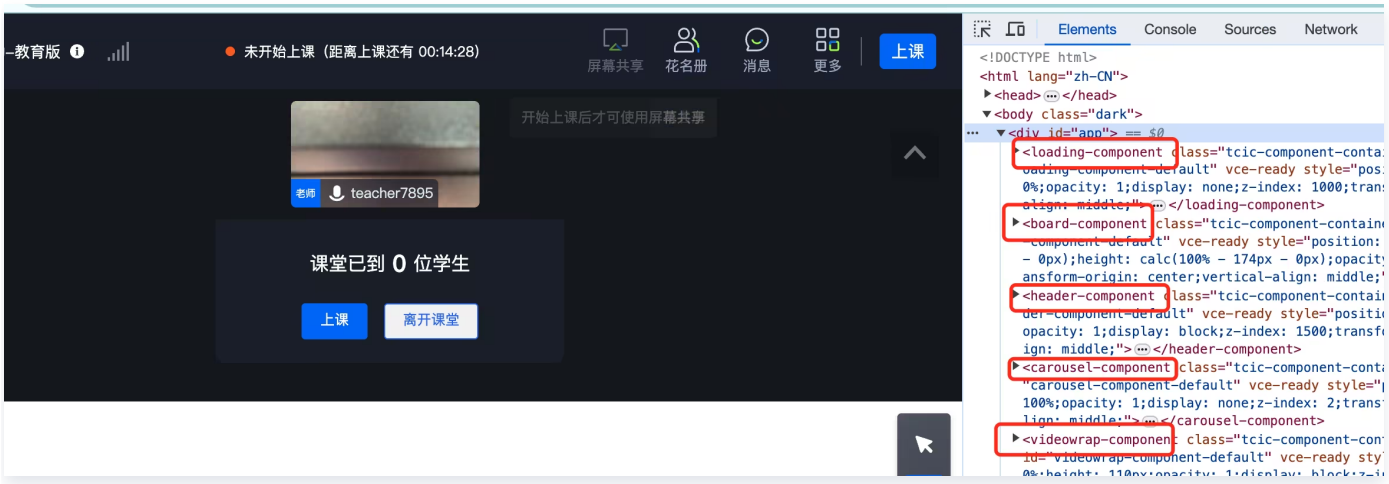


同样的方式可以用于屏蔽更多内容。

> **注意：**
> 使用 CSS 强制覆盖前，建议仔细观察样式名称，避免屏蔽预期外的元素。

示例二：结合 js 临时屏蔽模块

在示例一中，可以看到我们有些业务组件有特殊名称。如图所示，这些特殊名称可以帮助我们快速定位业务组件位置。



假设我们希望在用户操作时自动隐藏某些组件，而在不操作时则显示这些组件。这时仅仅通过 CSS 无法实现，您可以在 快速开始定制 创建的 `test.js` 中粘贴以下代码来实现。

```
/**
 * 注入JS会在全局环境下找到TCIC对象
 * TCIC.SDK.instance 为SDK实例
 * TCIC.TMainState 为SDK状态枚举
 * Joined_Class 已加入课堂，此时组件都被加载可以使用
```

```
 * promiseState 表示当状态Joined_Class为true时，必定会执行一次
 */
TCIC.SDK.instance.promiseState(TCIC.TMainState.Joined_Class, true).then(()
=> {
  console.log("%c [customJS] willHide", "font-size:16px;color:red");
  /**
   * 这些是上图所示的组件名称
   */
  let hideList = ["header-component", "videowrap-component"];
  let timer = null;
  /**
   * 监听鼠标移动事件，用户移动鼠标就隐藏组件
   */
  document.body.addEventListener("mousemove", () => {
    hideList.forEach((item) => {
      TCIC.SDK.instance.getComponent(item).getVueInstance().hide();
    });
    clearTimeout(timer);
    /**
     * 用户停止移动鼠标1秒后，显示组件
     */
    timer = setTimeout(() => {
      hideList.forEach((item) => {
        TCIC.SDK.instance.getComponent(item).getVueInstance().show();
      });
      timer = null;
    }, 1000);
  });
});
```

其它文档

更多 SDK 方法请参考 TCIC-SDK API 文档 。

不同角色的下课倒计时提醒

最近更新时间：2026-03-18 16:41:01

> **说明：**
> 阅读本文前，请确保您已经了解 快速开始定制 的内容。

功能描述

区分角色，针对老师、巡课、助教、学生在距离下课还有「x分钟」时，进行弹窗提醒。

页面展示

弹窗文案仅供参考。

操作方法

粘贴以下代码到 快速开始定制 里准备好的 `test.js` 中，即可实现下课倒计时提醒。
以下案例为：距离下课5分钟时，老师侧进行弹窗提醒。

```
TCIC.SDK.instance
  .promiseState(TCIC.TMainState.Class_Status,
TCIC.TClassStatus.Already_Start)
  .then(() => {
    const classInfo = TCIC.SDK.instance.getClassInfo();
    const { endTime, startTime } = classInfo;
    const now = Date.now();
    // 提前 5 分钟弹窗
    /**
     * note: 您也可以通过订阅 TCIC.TMainState.Class_Duration 来获取课程进行时
间，做些定制化处理
     * TCIC.SDK.instance.subscribeState(TCIC.TMainState.Class_Duration,
console.warn)
     */
```

```
    const timeToAlert = (endTime - 5 * 60) * 1000;
    if (TCIC.SDK.instance.isTeacher() && now <= timeToAlert) {
      setTimeout(() => {
        TCIC.SDK.instance.showMessageBox('温馨提示', '本堂课将于 5 分钟后结束，
请老师合理安排时间', ['确定'] , () => console.log('弹窗关闭'));
      }, timeToAlert - now);
    } else {
      TCIC.SDK.instance.showMessageBox('温馨提示', '本堂课即将结束，请老师合理安
排时间', ['确定'] , () => console.log('弹窗关闭'));
    }

  });
```

麦克风及摄像头开关控制（学生侧）

最近更新时间：2026-03-18 16:41:02

为优化业务侧在不同教学场景的交互使用体验，目前支持配合后台 API 接口 创建房间 的参数和前端自定义方法，实现课中对学生麦克风、摄像头权限的精细化控制。

> **说明：**
> 阅读本文前，请确保您已经了解 快速开始定制 的内容。

前端自定义 JS 方法

通过以下代码即可控制上台学生的麦克风/摄像头的开/关状态：

```
TCIC.SDK.instance.setStageUpMediaOption({
        micAutoOpen: true,
        cameraAutoOpen: false,
  });
```

具体场景使用说明示例

`EnableDirectControl` 含义：是否允许老师/助教直接控制学生的摄像头/麦克风（是否需要学生授权同意）。
该设置适用于以下场景：

- 当老师邀请学生上台时；
- 当老师播放音视频课件时。

自定义 JS 内容含义：无需授权的情况下，麦克风和摄像头的开启/关闭状态。

接口配置	同时：前端自定义 JS 内容	学生侧-交互表现
设置创建房间接口的 `EnableDirectControl=0`	micAutoOpen: false, cameraAutoOpen: false,	麦克风：展示弹窗以进行授权 摄像头：展示弹窗以进行授权
设置创建房间接口的 `EnableDirectControl=0`	micAutoOpen: false, cameraAutoOpen: true,	麦克风：展示弹窗以进行授权 摄像头：默认打开
设置创建房间接口的 `EnableDirectControl=0`	micAutoOpen: true, cameraAutoOpen: false,	麦克风：默认打开 摄像头：展示弹窗以进行授权
仅需设置创建房间接口的 `EnableDirectControl=1`	micAutoOpen: false,	麦克风：默认关闭 摄像头：默认关闭

	cameraAutoOpen: false,	
设置创建房间接口的 `EnableDirectControl=1`	micAutoOpen: true, cameraAutoOpen: true,	麦克风：默认打开 摄像头：默认打开
设置创建房间接口的 `EnableDirectControl=1`	micAutoOpen: true, cameraAutoOpen: false,	麦克风：默认打开 摄像头：默认关闭
设置创建房间接口的 `EnableDirectControl=1`	micAutoOpen: false, cameraAutoOpen: true,	麦克风：默认关闭 摄像头：默认打开

调整移动端音量类型（Web 内核）

最近更新时间：2026-03-18 16:41:02

> **说明：**
> 阅读本文前，请确保您已经了解 快速开始定制 的内容。

功能描述

现代智能手机中通常都具备两套系统音量类型，即“通话音量”和“媒体音量”。通过自定义本功能，您可以选择当前 App 使用哪种音频通道。

- 通话音量：手机专为接打电话设计的音量类型，具备回声抵消（AEC）功能，支持通过蓝牙耳机上的麦克风进行拾音，但音质相对普通。

  当用户通过手机侧面的音量按键调低音量时，如果无法将其调至零（也就是无法彻底静音），表明您的手机当前处于通话音量类型。
- 媒体音量：手机专为音乐场景设计的音量类型，无法使用系统的 AEC 功能，并且不支持通过蓝牙耳机的麦克风进行拾音，但具备更好的音乐播放效果。

  当用户通过手机侧面的音量按键调低音量时，如果能够将手机音量调至彻底静音，表明您的手机当前处于媒体音量类型。

SDK 目前提供了三种系统音量类型的控制模式：

- 自动切换模式
- 全程通话音量模式
- 全程媒体音量模式

默认情况下，iOS 端使用自动切换模式，而 Android 端则采用全程媒体音量模式。

自定义方法

> **说明：**
> 移动端 SDK 版本要求：
> - iOS 端：1.8.5.14及以上，可参见 Web 内核（iOS）快速接入 。
> - Android 端：1.8.19及以上，可参见 Web 内核（Android）快速接入 。

在 JS 文件中 `promiseState TCIC.TMainState.Joined_TRTC` 设置为 `true`，即音视频初始化并进入完成后，再调用 `TCIC.SDK.instance.setSystemVolumeType` 方法配置系统音量类型。具体参数说明：

取值	描述
0	自动切换模式。
1	全程媒体音量模式。

2	全程通话音量模式。

参考代码：

```
TCIC.SDK.instance.promiseState(TCIC.TMainState.Joined_TRTC, true)
          .then(() => {
            TCIC.SDK.instance.setSystemVolumeType(0);
          });
```

修改白板文本工具的字体大小

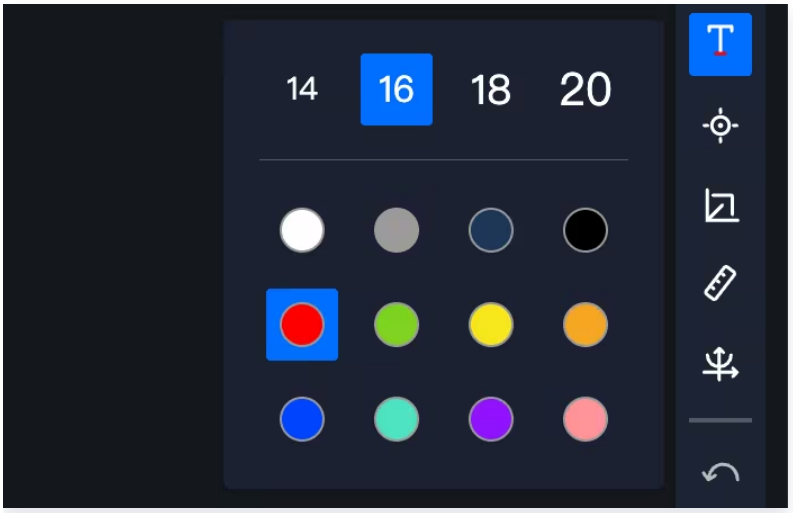
最近更新时间：2026-03-18 16:41:02

> **说明：**
> 阅读本文前，请确保您已经了解 快速开始定制 的内容。

功能描述

场景

课堂白板的文本工具默认四个字体大小值：14、16、18、20。 每一个值对应映射不同的实际渲染字体大小值，分别对应280，320，360，400。用户在白板上书写文字的大小, 实际由映射的实际值来决定。



您可以修改该四个默认字体大小值的对应实际映射渲染字体大小值，来修改文本工具的实际字体大小。也可以新增或减少字体大小值，使得用户能够更精准管控。

操作方法

仅修改默认渲染字体大小值

粘贴以下代码到 快速开始定制 里准备好的 `test.js` 中，即可完成修改。具体说明请见注释。

```
TCIC.SDK.instance.promiseState(TCIC.TMainState.Joined_Class, true).then(()
=> {
 TCIC.SDK.instance.promiseState(TCIC.TMainState.Board_Ready, true).then(()
=> {
 const tool = TCIC.SDK.instance.getComponent('board-tool-
component').getVueInstance();
 tool.textSize.sizes = [280, 320, 360, 400]; // 这里分布对应工具默认四个字体大小
值: 14 16 18 20
```

```
 TCIC.SDK.instance.getBoard().setTextSize(第二个字体16的size, 上面第二项的值,
用于默认);
 });
});
```

新增或减少字体的大小值

粘贴以下代码到 快速开始 里准备好的 `test.js` 中，即可完成修改。具体说明请见注释。

```
TCIC.SDK.instance.promiseState(TCIC.TMainState.Joined_Class, true).then(()
=> {
 TCIC.SDK.instance.promiseState(TCIC.TMainState.Board_Ready, true).then(()
=> {
 const tool = TCIC.SDK.instance.getComponent('board-tool-
component').getVueInstance();
 tool.textSize.values = [14, 16, 18, 20, 22, 24]; // 所有需要对用户展示的字体大
小值
 tool.textSize.sizes = [280, 320, 360, 400, 600, 750]; // 上述所有值的映射实际
大小值. 该数组每一项均和 values 数组每一项对应
 TCIC.SDK.instance.getBoard().setTextSize(上面第二项的size值, 用于默认);
 });
});
```

此外，再自定义 css 指定如下样式：

```
.board-drop-list.popper-text i {
    font-size: 16px !important;
}
```

显示字幕转写开关

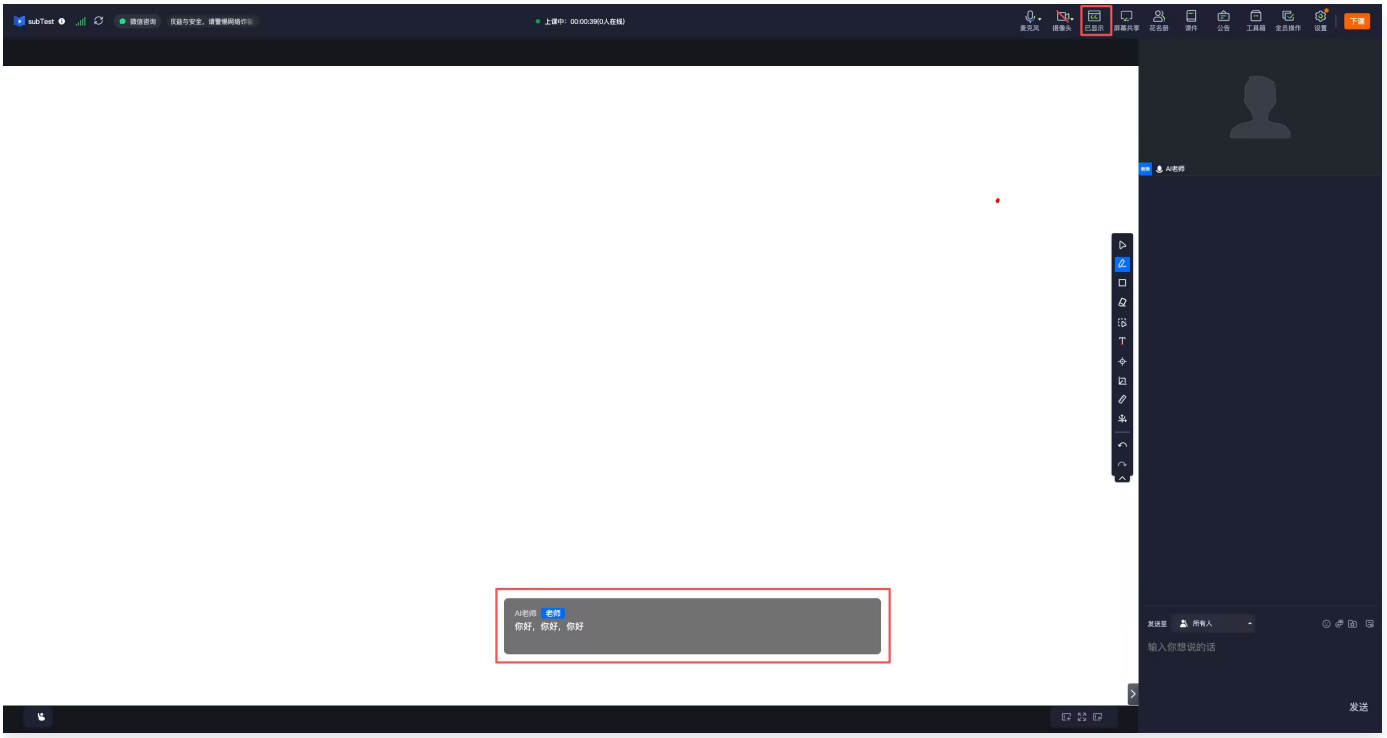
最近更新时间：2026-03-18 16:41:02

> **说明：**
> 阅读本文前，请确保您已经了解 快速开始定制 的内容。

功能描述

在上课过程中，用户可以通过开启字幕转写，将实时语音转写成字幕内容并展示在课堂界面中，且支持灵活拖动。当有用户讲话时，展示实时字幕。若所有用户停止说话超过 5S，则实时字幕 UI 关闭（字幕转写功能仍保持开启，当有用户再次说话时，实时字幕会再次展示）。

效果展示



操作方法

1. 调用 创建课堂 API 时，设置为手动转写模式（ `SubtitlesTranscription = 2` ）。老师或助教进入课堂后，将显示字幕转写的开关，用户可以按需开启，即可展示课中讲话时对应的实时字幕。
2. 如果想自定义配置实时字幕功能权限，可粘贴以下代码到 快速开始定制 里准备好的 `test.js` 中，即可实现自定义配置字幕转写权限。

例如，关闭助教的字幕转写权限，如下代码案例：

```
TCIC.SDK.instance.promiseState(TCIC.TMainState.Class_Info_Ready,
true).then(() => {
```

```
  if (TCIC.SDK.instance.isAssistant()) {
    TCIC.SDK.instance.setFeatureAvailable('SubtitleTranscription', false);
  }
})
```

配置白板功能权限

最近更新时间：2026-03-18 16:41:01

> **说明：**
> 阅读本文前，请确保您已经了解 快速开始定制 的内容。

功能描述

场景

课中老师邀请学生上台后，需要给学生开启白板权限。不同行业场景里，学生需要的白板功能不同，故支持客户根据应用 id 维度自定义配置功能（对应前端隐藏操作入口）。

默认配置

白板权限开启后，新增/删除白板、涂鸦权限、课件操作（翻页、放大/缩小）功能都开启。

页面位置

操作方法

粘贴以下代码到 快速开始定制 里准备好的 `test.js` 中，即可实现配置白板功能权限。
例如，仅开启学生在白板上的涂鸦权限，如下代码案例：

```
TCIC.SDK.instance.promiseState(TCIC.TMainState.Class_Status,
TCIC.TClassStatus.Already_Start).then(() => {
        const isStudent = TCIC.SDK.instance.isStudent();
        if (isStudent) {
            TCIC.SDK.instance.setFeatureAvailable('WhiteBoardPPT', false);

TCIC.SDK.instance.setFeatureAvailable('WhiteBoardPPT.WheelPaging', false);
            TCIC.SDK.instance.setFeatureAvailable('WhiteBoardList',
false);
            const teduBoard = TCIC.SDK.instance.getBoard();
            TCIC.SDK.instance.promiseState(TCIC.TMainState.Board_Ready,
true).then(() => {
                console.log('TCIC.TMainState.Board_Ready');
                setTimeout(() => {

teduBoard.enablePermissionChecker(['Board::Switch::Step'], ['operator/']);

teduBoard.enablePermissionChecker(['Board::Switch::Page'], ['operator/']);
                }, 500);
            });
        }
      });


      TCIC.SDK.instance.subscribeState(TCIC.TMainState.Board_Permission,
async (value) => {
        const isStudent = TCIC.SDK.instance.isStudent();
        if (isStudent && value) {
            await
TCIC.SDK.instance.promiseState(TCIC.TMainState.Board_Ready, true);
           setTimeout(() => {
            const teduBoard = TCIC.SDK.instance.getBoard();
            teduBoard.enablePermissionChecker(['Board::Switch::Step'],
['operator/']);
            teduBoard.enablePermissionChecker(['Board::Switch::Page'],
['operator/']);
           }, 500);
        }
    });
```

过滤进出课堂的消息提示

最近更新时间：2025-12-12 11:38:11

在实时互动-教育版的课堂中，当有用户进入或离开课堂时，系统会默认显示相应的提示消息。在某些场景下，业务侧可能需要对这些消息进行过滤或自定义控制，以提供更好的用户体验。
本文档将介绍如何使用 Web 端 SDK 提供的 API 来灵活控制进出课堂的消息显示。

功能特性

- 全局控制：一键开启或关闭所有进出课堂的消息。
- 精准过滤：基于用户角色进行精准控制消息展示。

实现方法

使用 setMemberJoinExitRoomInfoFilter 方法可以过滤用户进出课堂消息的显示。

- 基础用法

```
TCIC.SDK.instance.setMemberJoinExitRoomInfoFilter(function (msg) {
    console.log('进出课堂消息:', msg);
    if (msg.userId === 'your_user_id') {
        return false;
    }

    // return true 表示过滤掉（不显示）该消息
    // return false 表示显示该消息
    return true; // 过滤掉所有消息
});
```

- 基于用户角色的过滤

```
TCIC.SDK.instance.setMemberJoinExitRoomInfoFilter(({ msg }) => {
    try {
        // 解析消息数据
        const messageData = JSON.parse(msg.data);
        const userRole = messageData.data.data[0].role;

        // 角色说明:
        // 0 - 学生
        // 1 - 老师
        // 2 - 助教
        // 3 - 巡课
```

```
            // 过滤掉巡课的进出课堂消息
            if (userRole === 3) {
                return true;
            }

            return false; // 显示其他角色的消息
        } catch (error) {
            console.error('消息解析错误:', error);
            return false; // 解析失败时默认显示
        }
    });
```

新增购物车

最近更新时间：2025-08-20 17:45:21

购物车是实时互动-教育版面向营销场景的客户，提供的一个营销工具。老师（主播）可在课堂（房间）中单击对应的按钮，学生端(观众)可在弹出的购物车弹窗，进行后续相关操作。
下文提供了使用 Vue 实现购物车功能的方法，其他类似的功能如红包、答题器等功能也可以参考这个案例进行实现。
完整代码请单击 这里 查看。

开发老师（主播）端

我们通过 `useTask` 创建一个 Task，当老师调用 updateTask 时，学生端会收到老师端发来的购物车弹窗请求。

```
<template>
  <div class="custom-shop-btn" @click="showShopCart">
    <img src="../assets/shopcart.svg" alt="">
    <span class="header__btn-text">购物车</span>
  </div>
</template>

<script setup>
import useTask from '../hooks/useTask';
const { updateTask } = useTask('custom-shop-cart-tool');
const showShopCart = () => {
  updateTask({
    type: 'show-shop-cart',
  });
};
</script>

<style lang="less">
.custom-shop-btn {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  height: 100%;
  width: 60px;
  cursor: pointer;
  span{
    font-size: 12px;
```

```
    color: #a3aec7;
    line-height: 19px;
  }
  img{
    width: 24px;
    height: 24px;
  }
}
</style>
```

开发学生（观众）端

在学生端，我们监听 `custom-shop-cart-tool` 这个任务，当收到显示购物车的请求时，进行弹窗。

```
<template>
  <div class="shop-modal" v-if="showModal">
    <div class="shop-modal__content">
      <div class="shop-modal__header">
        <h2>购物车</h2>
        <button @click="showModal = false">关闭</button>
      </div>
      <div class="shop-modal__body">
        <p style="font-size: 30px;"></p>
        <p>1v1小班课</p>
        <p>优惠进行中</p>
      </div>
      <div class="shop-modal__footer">
        <button @click="showModal = false">取消</button>
        <button @click="goDetail()">查看详情</button>
      </div>
    </div>
  </div>
</template>

<script setup>
import { ref } from 'vue';
import useTask from '../hooks/useTask';

const showModal = ref(false);
useTask('custom-shop-cart-tool', (data) => {
  // 任务更新回调
```

```
  if (data.type === 'show-shop-cart') {
    // 展示购物车
    if (localStorage.getItem('hasShown') === 'true') {
      return;
    }
    TCIC.SDK.instance.promiseState('TStateDeviceDetect', false).then(() =>
{
      // 设备检测完成后展示弹窗
      showModal.value = true;
      localStorage.setItem('hasShown', 'true');
    });
  }
});
const goDetail = () => {
  showModal.value = false;
  // 自行实现跳转到详情页
};
</script>
```

说明：

完成开发后，我们可以通过 场景配置 将打包后的 JS 上传，并在创建该场景下的课堂中自动加载自定义JS。

新增签到功能

最近更新时间：2025-04-10 17:03:12

签到是课堂教学中的一个通用功能，老师可在课堂中单击对应的按钮，学生端可在弹出的签到弹窗上单击按钮完成签到。
下文提供了使用 Vue 实现签到功能的方法，完整代码请点击 这里 查看。

开发老师端

我们通过 `useTask` 创建一个 Task，当老师调用 updateTask 时，学生端会收到老师端发来的签到请求。

```
<template>
  <div class="custom-checkin-btn" @click="checkIn">
    <span class="header__btn-text">签到</span>
  </div>
</template>

<script setup>
// 创建一个签到任务 custom-check-in-tool
const { updateTask } = useTask('custom-check-in-tool');
// 点击签到按钮时，发起签到
const checkIn = () => {
  updateTask({
    type: 'ask-check-in',
  });
};
</script>

<style lang="less">
...
</style>
```

开发学生端

在学生端，我们监听 `custom-check-in-tool` 这个任务，当收到签到请求时，进行弹窗，让学生完成签到。

```
<template>
  <div class="checkin-modal" v-if="showModal">
    <div class="checkin-modal__content">
      <div class="checkin-modal__header">
```

```
        <h2>签到</h2>
        <button @click="showModal = false">关闭</button>
      </div>
      <div class="checkin-modal__body">
        <p>请您完成签到</p>
      </div>
      <div class="checkin-modal__footer">
        <button @click="showModal = false">取消</button>
        <button @click="handleCheckIn()">确认签到</button>
      </div>
    </div>
  </div>
</template>

<script setup>
import { ref } from 'vue';
import useTask from '../hooks/useTask';

const showModal = ref(false);
const isCheckIn = ref(false);

// 监听到老师发起签到任务的回调函数
useTask('custom-check-in-tool', (data) => {
  if (data.type === 'ask-check-in') {
    // 收到签到请求
    console.log('收到签到请求');
    // 检查是否已签到，没有签到就展示签到弹窗
    if (!isCheckIn.value) {
      TCIC.SDK.instance.promiseState('TStateDeviceDetect', false).then(()
=> {
          // 设备检测完成后展示弹窗
          showModal.value = true;
        });
    } else {
      // 已经签到，提示用户
      console.log('已经签到');
    }
  }
});
const handleCheckIn = () => {
  isCheckIn.value = true;
```

```
  showModal.value = false;
  // 发送请求到服务器
};
</script>

<style lang="less">
...
</style>
```

说明：

完成开发后，我们可以通过 场景配置 将打包后的 js 上传，创建该场景的课堂后，课堂将自动加载自定义 js。

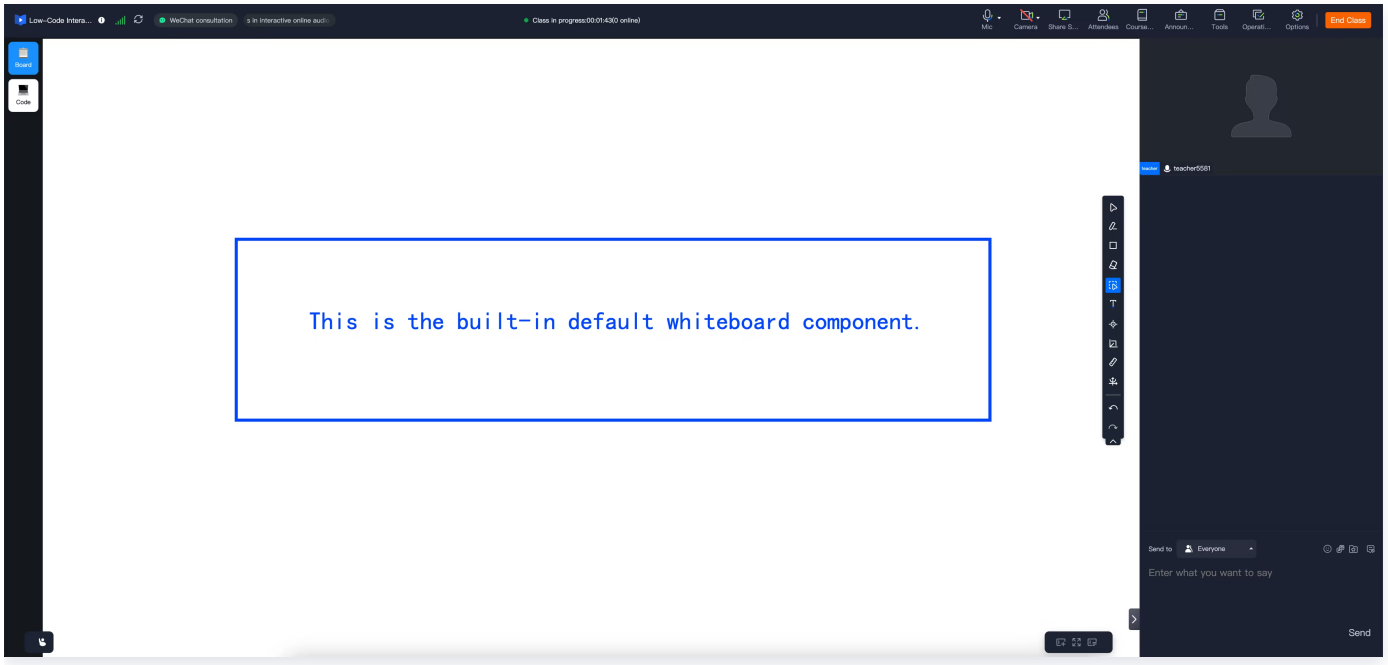
多层级可切换 Tab 白板区域

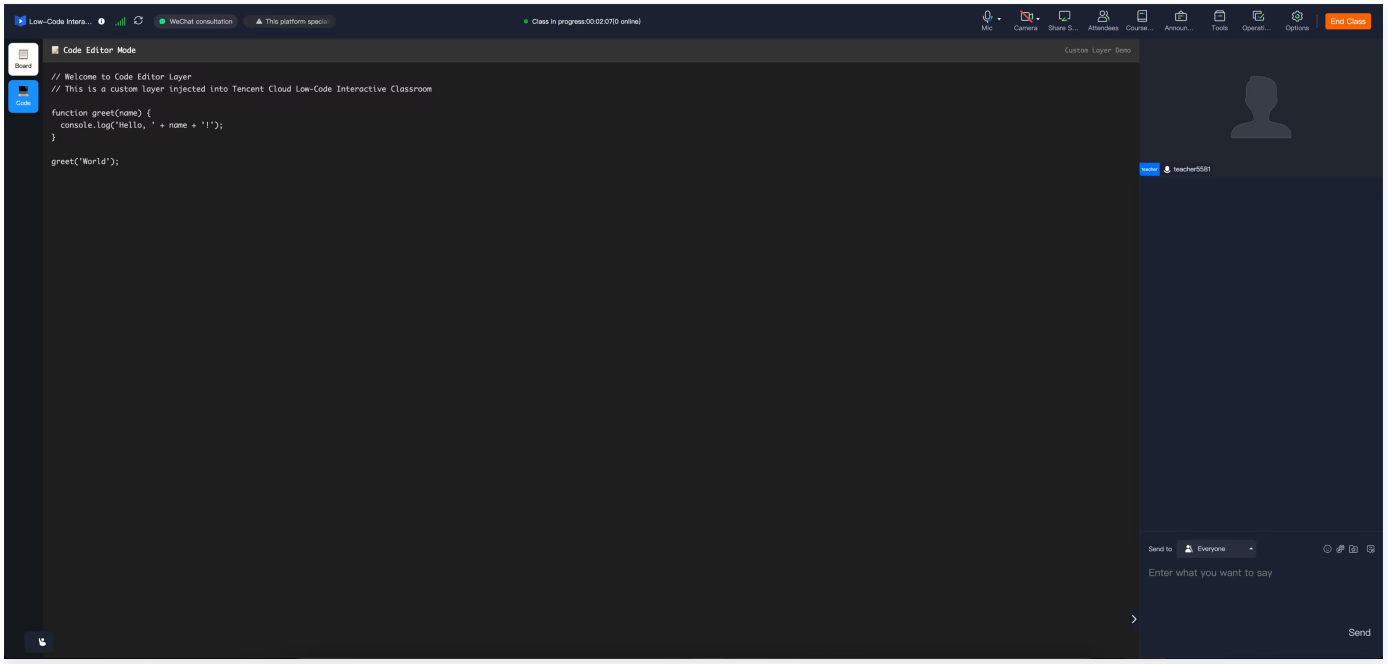
最近更新时间：2026-03-18 16:41:02

> **说明：**
> 阅读本文前，请确保您已经了解 Web 及 H5 快速接入 的内容。

功能描述

本文档将介绍如何在互动课堂 SDK 的白板区域集成自定义的多层级视图（如代码编辑器、文档预览等），并通过自定义的侧边 Tab 实现层级切换。
演示效果如下图所示（左侧切换器与中间多层级区域）。请注意，该区域的 UI 均支持自定义，您需要自行开发 UI 组件传入。因此下图样式仅供参考。





> **说明：**
> 为方便您接入，上方效果图的 Demo 演示项目可参考 **Demo GitHub 仓库**。您可基于此 Demo 的开源代码，实现本功能的自定义开发。

核心机制

我们提供插槽机制，允许开发者通过以下两个状态注入自定义 UI：

- `TStateCustomLayers`：定义除默认白板外的其他自定义层级（Layer）。通过数组形式指定并管理多个层级。
- `TStateLayerSwitcherTab`：定义显示在白板左侧工具栏的切换器组件（Tab）。开发者可以通过调用 `activateLayer` 方法来控制不同层级的显示与隐藏（课堂会自动管理层级堆叠顺序）。

注意事项

1. **样式隔离与背景色**
   - 自定义层级组件（如 `my-code-editor`）**必须设置背景色**（如 `background: #fff`），否则会透出下方的白板内容，造成视觉重叠。
   - 自定义层级组件（如 `my-code-editor`）**必须配置高度及宽度为 100%**，以确保占满提供的 Dom 节点挂载区域。
2. **组件注册时机**

   请确保在执行 `setState` 配置之前，`customElements.define` 已经执行完毕，否则 SDK 渲染时可能无法识别组件标签。
3. **生命周期管理**

   层级切换本质上是 CSS `z-index` 的变化。Vue 组件**不会被销毁**，其内部状态（如编辑器中的代码内容）会保持。

开发步骤

步骤一：开发自定义组件

为了确保组件样式的独立性并避免框架版本冲突，推荐使用 Vue 3 的 `defineCustomElement` 将业务组件封装为标准 Web Component。

1. 定义若干个业务层级组件 (例如：CodeEditorElement.ce.vue)

```
<!-- src/CodeEditorElement.ce.vue -->
<template>
  <div class="code-editor-layer">
    <div class="toolbar">代码编辑器模式</div>
    <textarea class="editor-area" placeholder="在此输入代
码...">console.log('Hello World');</textarea>
  </div>
</template>

<script setup>
// 组件逻辑...
</script>

<style>
.code-editor-layer {
  width: 100%;
  height: 100%; /* 务必占满容器 */
  background: #1e1e1e; /* 务必设置背景色，防止透出下方白板 */
  display: flex;
  flex-direction: column;
}
/* ...其他样式 */
</style>
```

2. 定义切换器组件 (例如：LayerSwitcherComponent.ce.vue)

切换器负责调用 SDK API 切换视图。

```
<!-- src/LayerSwitcherComponent.ce.vue -->
<template>
  <div class="layer-switcher">
    <button @click="switchLayer('whiteboard')">白板</button>
    <button @click="switchLayer('code-editor')">代码</button>
  </div>
```

```
</template>

<script setup>
/**
 * 切换层级方法
 * @param {string} layerId - 目标层级ID ('whiteboard' 为默认内嵌白板组件)
 */
const switchLayer = (layerId) => {
  TCIC.SDK.instance.getComponent('board-
component').getVueInstance().activateLayer(layerId);
};
</script>
```

步骤二：注册 Web Component

在项目入口文件（如 `main.js`）中，将上述组件注册为浏览器可识别的 Custom Elements。

```
import { defineCustomElement } from 'vue';
import CodeEditorElement from './CodeEditorElement.ce.vue';
import LayerSwitcherComponent from './LayerSwitcherComponent.ce.vue';

// 1. 注册代码编辑器组件
// 注册名为 'my-code-editor'
customElements.define('my-code-editor',
defineCustomElement(CodeEditorElement, { shadowRoot: false }));

// 2. 注册切换器组件
// 注册名为 'my-layer-switcher'
customElements.define('my-layer-switcher',
defineCustomElement(LayerSwitcherComponent, { shadowRoot: false }));
```

步骤三：配置 SDK 状态

步骤二组件注册完成后，通过 `setState` 将注册好的组件标签名注入到课堂前端项目中。

```
// 获取 SDK 实例
const sdkInstance = TCIC.SDK.instance;

// 1. 配置自定义层级列表
// 请注意，id 请不要占用我们内嵌默认白板的 "whiteboard"
const customLayersConfig = [
```

```
  {
    id: 'code-editor',        // 层级唯一标识 (Layer ID)，用于 activateLayer 调
用
    component: 'my-code-editor' // 上一步注册的 Web Component 标签名
  }
  // 可配置多个...
];

// 注入自定义层级配置
sdkInstance.setState('TStateCustomLayers', customLayersConfig);

// 2. 注入侧边切换 Tab 组件
sdkInstance.setState('TStateLayerSwitcherTab', 'my-layer-switcher');
```

API 参考

方法：`activateLayer(layerId)`

切换当前激活的层级。SDK 会自动处理层级的 `z-index`（激活层级置顶）。

- **参数：**

  `layerId`：String 类型。
  - `'whiteboard'`：固定值，代表默认的互动白板层级。
  - 自定义 ID：对应 `TStateCustomLayers` 配置中的 `id`（例如 `'code-editor'`）。
- **返回：** `Boolean`（执行是否成功）

示例：

```
const boardInstance = TCIC.SDK.instance.getComponent('board-
component').getVueInstance();

// 切换到代码编辑器
boardInstance.activateLayer('code-editor');
// 切换回白板
boardInstance.activateLayer('whiteboard');
```

原生内核定制指南
Android 定制指南（原生内核）

最近更新时间：2026-03-18 16:41:02

> **说明：**
> 阅读本文前，请确保您已经了解并完成了 接入准备 和 原生内核（Android）快速接入 的基础集成。

SDK 支持高度自定义课堂内的 UI（如头部导航栏、消息气泡等）。以下以**自定义头部左侧组件**为例。

1. 创建自定义 View (Java 示例)

实现 `NativeViewCreator` 接口，编写您的原生 Android View：

```
public class HeaderLeftViewCreator implements NativeViewCreator {
    @Override
    public View createView(Context context, int id, Object args) {
        // 创建一个水平布局
        LinearLayout layout = new LinearLayout(context);
        layout.setOrientation(LinearLayout.HORIZONTAL);
        layout.setGravity(Gravity.CENTER_VERTICAL);

        // 添加自定义文字
        TextView textView = new TextView(context);
        textView.setText("自定义头部");
        textView.setTextColor(Color.WHITE);
        layout.addView(textView);

        // 添加自定义图标
        ImageView infoIcon = new ImageView(context);

infoIcon.setImageResource(android.R.drawable.ic_menu_info_details);
        layout.addView(infoIcon);

        return layout;
    }

    @Override
    public void disposeView(View view) {
        // 页面销毁时的资源清理工作
```

```
    }
}
```

2. 将自定义 View 注册到配置中

在进房前，将写好的 Creator 赋值给 `TCICConfig`：

```
// 1. 实例化头部配置
val headerConfig = TCICHeaderComponentConfig()
// 2. 绑定自定义 View
headerConfig.setHeaderLeftBuilder(::HeaderLeftViewCreator)

// 3. 写入全局配置并进房
val config = TCICConfig(token, classId, userId, role)
config.headerComponentConfig = headerConfig
TCICManager.setConfig(config)

context.startActivity(TCICManager.getTCICIntent(context))
```

附：支持自定义的组件列表

您可以根据业务需求，对以下模块进行自定义替换：

- 头部组件（`TCICHeaderComponentConfig`）：左侧视图、中间视图、右侧视图、操作按钮。
- 消息组件（`TCICMessageComponentConfig`）：消息列表头部、消息气泡、单条消息 Item。
- 音视频组件（`TCICVideoComponentConfig`）：视频操作栏、悬浮视频窗口。

iOS 定制指南（原生内核）

最近更新时间：2026-03-18 16:41:02

> **说明：**
> 阅读本文前，请确保您已经了解并完成了 接入准备 和 原生内核（iOS）快速接入 的基础集成。

SDK 支持将原生的 `UIView` 嵌入到课堂 UI 中。

1. 创建 Native View

继承 `TCICViewFactory` 并实现视图构建逻辑：

```
import Foundation
import UIKit
import tcic_ios
import Flutter

class MyHeaderLeftView: TCICViewFactory {
    override init(messenger: FlutterBinaryMessenger) {
        super.init(messenger: messenger)
    }

    override func createNativeView(frame: CGRect, viewId: Int64, args:
Any?) -> UIView {
        let view = UIView(frame: frame)
        view.backgroundColor = .blue

        let label = UILabel(frame: CGRect(x: 0, y: 0, width: 180, height:
48))
        label.text = "My Header Left View From iOS"
        label.textColor = .white
        label.textAlignment = .center
        view.addSubview(label)

        return view
    }
}
```

2. 将 Native View 注册到 SDK

在配置 `TCICConfig` 时，将自定义 View 传入对应的 ComponentConfig 中：

```
let headerConfig = TCICHeaderComponentConfig()

let headerLeftBuilder: TCICHeaderComponentConfig.HeaderBuilder = {
    return MyHeaderLeftView(messenger:
TCICManager.shared.Tengine.binaryMessenger)
}

headerConfig.headerLeftBuilder = headerLeftBuilder // 绑定 Builder
headerConfig.headerLeftBuilderWidth = 200          // 设置宽度
headerConfig.headerLeftBuilderHeight = 40          // 设置高度

let config = TCICConfig(
    token: "YOUR_TOKEN",
    classId: "YOUR_CLASS_ID",
    userId: "YOUR_USER_ID",
    role: 1,
    headerComponentConfig: headerConfig
)
TCICManager.shared.setConfig(config)
```

API 参考

TCICConfig 配置类

基础配置

```
public class TCICConfig {
    let token: String              // 用户认证令牌
    let classId: String            // 课程 ID
    let userId: String             // 用户 ID
    let role: Int                  // 用户角色
    let langConfig: String?        // 语言配置
    let isLatestBackend: Bool      // 是否使用最新后端环境
    let isTestBackend: Bool        // 是否使用测试后端环境
}
```

组件配置

```
public class TCICConfig {
    let headerComponentConfig: TCICHeaderComponentConfig?          // 头部组
件配置
    let messageComponentConfig: TCICMessageComponentConfig?        // 消息组
件配置
    let videoComponentConfig: TCICVideoComponentConfig?            // 视频组
件配置
    let fontConfig: TCICFontConfig?                                // 字体配
置
    let settingComponentConfig: TCICSettingComponentConfig?        // 设置组
件配置
    let whiteBoardComponentConfig: TCICWhiteBoardComponentConfig? // 白板组
件配置
    let membersComponentConfig: TCICMembersComponentConfig?        // 成员组
件配置
}
```

回调配置 (TCICCallback)

```
public class TCICCallback {
    /// 加入课堂成功回调
    public var onJoinedClassSuccessBlock: (() -> Void)?

    /// 退出课堂回调
    public var afterExitedClassBlock: (() -> Void)?

    /// 加入课堂失败回调
    public var onJoinedClassFailedBlock: (() -> Void)?

    /// 被踢出课堂回调
    public var onKickedOffClassBlock: (() -> Void)?

    /// 成员加入课堂回调 （参数为用户信息字典）
    public var onMemberJoinedClassBlock: (([String: Any]) -> Void)?

    /// 成员离开课堂回调 （参数为用户信息字典）
    public var onMemberLeaveClassBlock: (([String: Any]) -> Void)?

    /// 收到消息回调 （参数为消息内容字典）
    public var onReceivedMessageBlock: (([String: Any]) -> Void)?
```

```
    /// 发生错误回调 （参数为：错误代码，错误信息）
    public var onErrorBlock: ((String, String) -> Void)?
}
```

组件配置系统

头部组件配置 (TCICHeaderComponentConfig)

```
let headerConfig = TCICHeaderComponentConfig()

// 设置左侧视图
headerConfig.headerLeftBuilder = {
    return MyHeaderLeftView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
headerConfig.headerLeftBuilderWidth = 200
headerConfig.headerLeftBuilderHeight = 40

// 设置头部主视图
headerConfig.headerBuilder = { ... }
headerConfig.headerBuilderWidth = 300
headerConfig.headerBuilderHeight = 50

// 设置头部操作视图
headerConfig.headerActionsBuilder = { ... }
headerConfig.headerActionsBuilderWidth = 150
headerConfig.headerActionsBuilderHeight = 40

// 设置右侧视图
headerConfig.headerRightBuilder = { ... }
headerConfig.headerRightBuilderWidth = 200
headerConfig.headerRightBuilderHeight = 40
```

消息组件配置 (TCICMessageComponentConfig)

```
let messageConfig = TCICMessageComponentConfig()

// 设置消息头部视图
messageConfig.messageHeaderBuilder = { ... }
```

```
messageConfig.messageHeaderBuilderWidth = 200
messageConfig.messageHeaderBuilderHeight = 30

// 设置消息气泡视图
messageConfig.messageBubbleBuilder = { ... }
messageConfig.messageBubbleBuilderWidth = 250
messageConfig.messageBubbleBuilderHeight = 80

// 设置消息项视图
messageConfig.messageItemBuilder = { ... }
messageConfig.messageItemBuilderWidth = 300
messageConfig.messageItemBuilderHeight = 60

// 设置消息行视图
messageConfig.messageRowBuilder = { ... }
messageConfig.messageRowBuilderWidth = 350
messageConfig.messageRowBuilderHeight = 40
```

音视频组件配置 (TCICVideoComponentConfig)

```
let videoConfig = TCICVideoComponentConfig()

// 设置视频操作视图
videoConfig.videoActionBuilder = {
    return MyVideoActionView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
videoConfig.videoActionBuilderWidth = 200
videoConfig.videoActionBuilderHeight = 50

// 设置视频浮动视图
videoConfig.videoFloatBuilder = {
    return MyVideoFloatView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
videoConfig.videoFloatBuilderWidth = 120
videoConfig.videoFloatBuilderHeight = 80
```

进阶功能
课堂生命周期说明

最近更新时间：2026-05-25 21:04:51

课堂状态

实时互动-教育版会有以下几种课堂状态：

> **说明：**
> 您可通过获取 房间配置信息 接口查询课堂状态。

课堂状态	参数值	说明

未开始	`Status` =0	还未到预约上课时间，且老师（或助教）没有点击开始**上课**。此时支持情况如下：
		• 老师（或助教）和学生均可以：
		  ○ 上课前设备检测、网络检测。
		  ○ 发送互动消息。
		  ○ 设置虚拟背景。
		• 仅老师（或助教）可以：
		  ○ 查看当前到课人数。
		  ○ 上传课件。
		  ○ 发布公告。
		  ○ 查看学生列表。
		  ○ 使用白板工具（效果不会同步学生侧）。
		  ○ 进行上课前其他设置（如是否显示学生举手、进出情况等）。

开始上课	`Status` =1	正式上课。开始统计课堂时长。 此时，老师（或助教）、学生均可以： • 音视频功能。 • 白板（涂鸦、展示课件、屏幕共享等）。 • 发送互动消息等。
结束课堂	`Status` =2	结束课堂。结束统计课堂时长。
课堂过期	`Status` =3	没有开课，当前已过下课时间。

切换课堂状态

- 正式上课时，统计“实际房间开始时间”。以下情况会正式上课：

类型	说明
用户行为	未到预约上课时间，老师（或助教）提前进入房间，点击开始**上课**。
	到了预约上课时间，老师（或助教）点击开始**上课**。
云 API 接口控制	通过 开始直播 接口，正式上课。

- **结束课堂时，统计“实际房间结束时间”。以下情况对应正式下课：**

类型	说明
在预定的课堂结束时间 `EndTime` 之前。 说明：该字段参见 创建课堂 。	老师（或助教）操作**下课**。
创建课堂 中 `EndDelayTime` 设置**不限制拖堂**时间。 注意：该字段非必填。课堂默认为不限制拖堂时间。	到了预约下课时间，老师（或助教）操作**下课**。
	到了预约下课时间，且没有成员在线，自动结束课堂。房间内仍有成员在线时，会持续上课。 注意：为避免影响正常上课，预约下课时间5分钟后，校验是否有成员在线。
创建课堂 中 `EndDelayTime` 设置**不能拖堂**时间。	到预约下课时间（无论课堂内是否有成员在线），自动结束课堂。
创建课堂 中 `EndDelayTime` 设置拖堂时间。	根据设置的拖堂时间，自动结束课堂。 **注意：** 结束课堂时间是以**预约下课时间** + 拖堂时间来计算的，如果实际会提前或推迟开课，请根据实际情况合理设置拖堂时间，避免影响。
云 API 接口控制	通过 结束直播 接口，正式下课。
伪直播推流结束，自动结束课堂。	-
RTMP 推流课程	需设置**拖堂时间**，或在推流结束后调用 结束直播 ，及时结束课堂

说明：

- 创建课堂 预定开始至预定结束时间不得超过24小时。
- 针对开课持续时长超过24小时的课程，会启动自动结束机制，以避免超长课程带来不必要的资源消耗。

统计课堂时长

课堂时长 = 实际房间结束时间 - 实际房间开始时间。

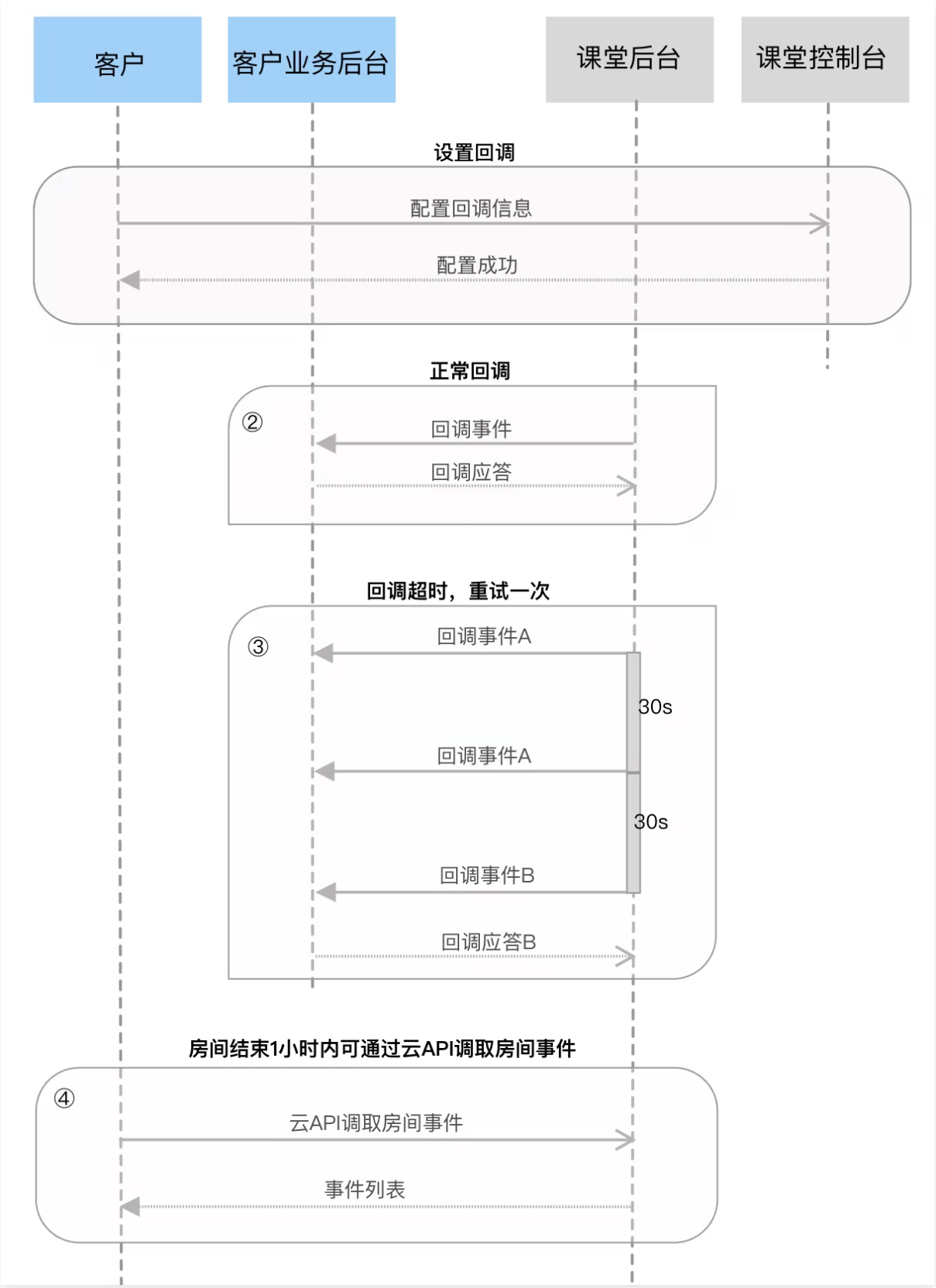
您可以通过 获取房间统计信息，查询课堂实际开始和实际结束时间，参见 RealStartTime 和 RealEndTime 。

服务端事件回调

最近更新时间：2026-06-01 19:54:31

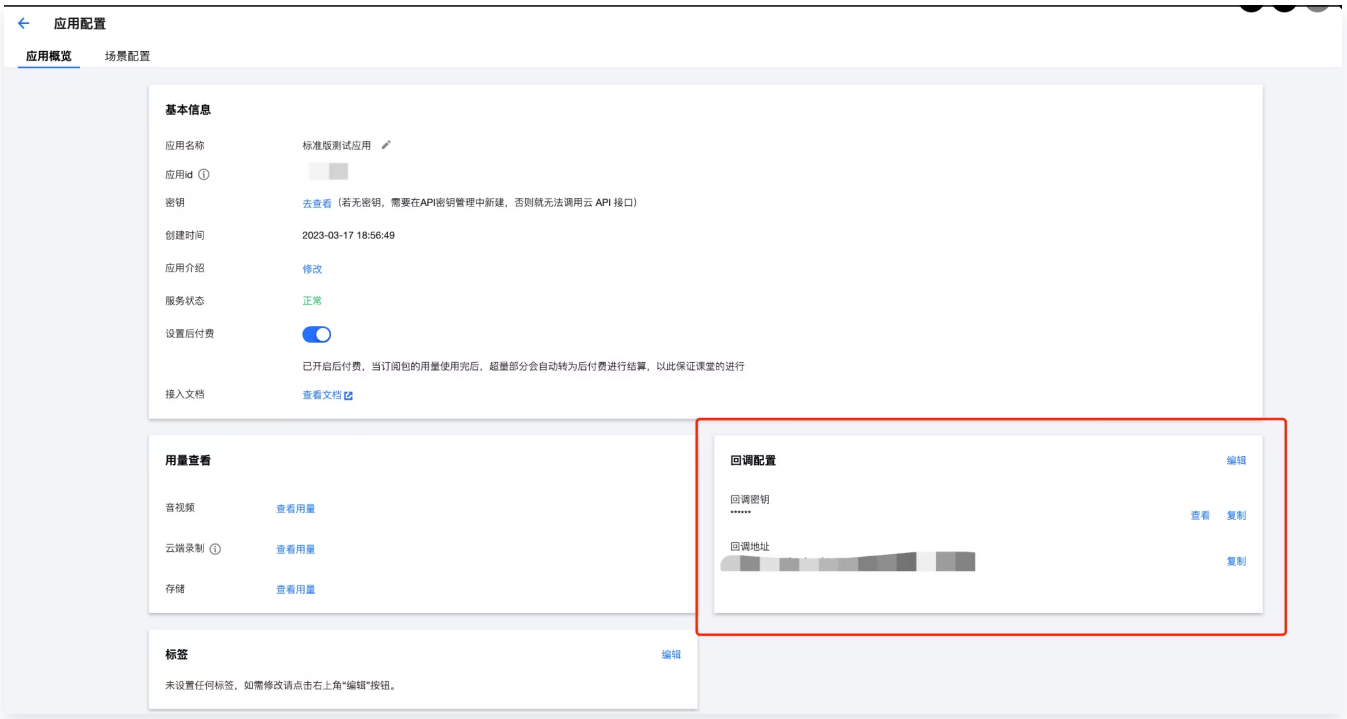
概述

> **说明：**
> 实时互动-教育版提供服务回调，支持将业务的状态通知回调至业务方，帮助业务方实现课前、课中以及课后的统计需求。



1. 回调地址配置。

   通过控制台 应用管理 中操作字段下的**应用配置**回调配置，完成回调地址（Callback）的配置。



2. 用户可以通过 获取房间事件接口 主动拉取房间事件。该接口仅支持房间结束1小时内拉取，过期房间事件会被释放。
3. 回调事件列表。

事件类型	事件名称	事件描述
房间事件	RoomStart	房间开始事件。
	RoomEnd	房间结束事件。
	RoomExpire	房间过期事件。
录制事件	RecordFinish	录制完成事件。
成员事件	MemberJoin	成员进入事件。
	MemberQuit	成员退出事件。
文档事件	DocumentTranscodeFinish	文档转码完成事件。
	DocumentCreate	文档创建事件。
	DocumentDelete	文档删除事件。
伪直播事件	FakeLiveStart	伪直播开始。
	FakeLiveStop	伪直播结束。
自定义事件	TaskUpdate	自定义事件。

转存事件	MixedFlowTransferStart	转存开始事件。
	MixedFlowTransferEnd	转存结束事件。
用量统计事件	MemberStatistics	房间用量统计事件。
板书截图事件	WhiteBoardSnapshotFinish	板书截图完成事件。
成员推断流事件	PushStream	成员开始推流事件。
	StopStream	成员停止推流事件。
页面录制事件	WebRecordFinish	页面录制异常结束事件。

回调签名方法

> ⚠ **注意：**
> 回调鉴权服务并非必须，但为了业务安全，建议业务层完成回调的鉴权校验。

对接回调鉴权部分，我们会提供 CallbackKey，用于对回调消息的鉴权。签名算法如下：

```
Sign = md5(CallbackKey+ExpireTime)

示例：
CallbackKey = Nj****Ey
ExpireTime = 1614151508
Sign = md5(Nj****Ey1614151508) = b9454ab5a85f9***************d34d
```

其中 ExpireTime 是签名过期时间，如果一条消息通知中的 ExpireTime 值所指定的时间已经过期，则可以判定这条通知无效，进而可以防止网络重放攻击。格式为十进制 UNIX 时间戳，即从1970年01月01日（UTC/GMT 的午夜）开始所经过的秒数。例如：

Go

```go
package main

import (
    "crypto/md5"
    "fmt"
)
```

```
func main() {
    callbackKey := "Nj****Ey"
    expire := 1614151508
    sign := fmt.Sprintf("%x", md5.Sum([]byte(callbackKey+fmt.Sprintf("%d",
 expire))))
    fmt.Printf(fmt.Sprintf("sign:%s", sign))    //
sign:b9454ab5a85f9***************d34d
}
```

Java

```
String md5Str =
DigestUtils.md5DigestAsHex("NjFGoDEy1614151508".getBytes());
```

Python3

```
# Python 3 code to demonstrate the
# working of MD5 (string - hexadecimal)

import hashlib

# initializing string
str2hash = "Nj****Ey1614151508"
result = hashlib.md5(str2hash.encode())

# printing the equivalent hexadecimal value.
print("sign:", end ="")
print(result.hexdigest())    #sign:b9454ab5a85f9***************d34d
```

在您收到回调请求时，会携带 ExpireTime 和 Sign 参数，您可以依据这两个值和回调密钥进行计算校验请求来源的正确性。

事件回调协议

我们会向注册的回调地址发起事件回调的形式是 HTTP POST 请求，请求体为 JSON 格式，内容为：

参数名称	类型	描述

Timestamp	Integer	事件生成的 Unix 时间戳，单位秒。
ExpireTime	Integer	签名的过期时间的 Unix 时间戳，单位秒，如果当前时间晚于过期时间，后台可以判断该请求不合法。
Sign	String	回调签名，可以根据事件回调鉴权中描述的方法校验签名是否匹配以校验该请求来源是否合法。
SdkAppId	Integer	本次事件所属的应用。
EventType	String	详见各回调类型描述。
EventData	JSON	详见各回调类型描述。

回调事件示例

```
POST xxxxxxxxxxxxxx(user-callback-url)
Content-Type: application/json; charset=utf-8
Accept: application/json

{
    "Timestamp":1679279232,
    "ExpireTime":1679279832,
    "Sign":"fbfd23733e626***************bc29",    // 参考 回调签名方法
    "SdkAppId":35***71,
    "EventType":"RoomStart",
    "EventData":{
        "RoomId":366317280
    }
}
```

我们会根据回调结果判断是否重试回调，回调应答HTTP状态200即为成功。

回调应答示例

应答： `HTTP STATUS CODE = 200`**，客户应答内容可携带** `JSON:` `{"error_code":0}`

```
HTTP STATUS CODE 200 OK

HTTP BODY:
{
    "error_code":0
}
```

事件列表

房间开始事件

事件类型

RoomStart

事件说明

当房间开始时发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。

示例

```
{
    "Timestamp":1679279232,
    "ExpireTime":1679279832,
    "Sign":"fbfd23733e626**************bc29",
    "SdkAppId":35***71,
    "EventType":"RoomStart",
    "EventData":{
        "RoomId":366317280
    }
}
```

房间结束事件

事件类型

RoomEnd

事件说明

当房间结束时发出通知。以下情况会触发结束房间。

字段描述	说明
EndReasonctient=0	老师/助教手动操作下课，结束课堂。
EndReasonApi=1	通过云 API 结束课堂。
EndReasonTimeOut= 2	到了下课时间结束课堂。此类情况仅针对：预约下课时间不可拖堂或明确拖堂时间的课堂。
EndReasonNoOnline = 3	房间内无人超时结束课堂。
EndReasonFakeLiveStop =4	伪直播课堂推流结束，自动结束课堂。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。

示例

```
{
    "Timestamp":1679279195,
    "ExpireTime":1679279795,
    "Sign":"696560af8fec9***************6448",
    "SdkAppId":35***71,
    "EventType":"RoomEnd",
    "EventData":{
        "RoomId":311601250
    }
}
```

房间过期事件

事件类型

RoomExpire

事件说明

当房间过期时发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。

示例

```
{
    "Timestamp":1679282220,
    "ExpireTime":1679282820,
    "Sign":"07e504e36a373***************0d3c",
    "SdkAppId":35***71,
    "EventType":"RoomExpire",
    "EventData":{
        "RoomId":310096990
    }
}
```

录制完成事件

事件类型

RecordFinish

事件说明

当录制完成生成下载地址时发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
Duration	Integer	录制时长，单位秒。
RecordUrl	Strin	录制地址（协议为 https）。

	g	
RecordSize	Integer	录制文件大小，单位 bit。

示例


```
{
    "Timestamp":1679279203,
    "ExpireTime":1679279803,
    "Sign":"7ada1f46f27ce***************e094",
    "SdkAppId":35***71,
    "EventType":"RecordFinish",
    "EventData":{
        "Duration":63,
        "RecordSize":698472,

"RecordUrl":"https://xxxxxxx.vod2.myqcloud.com/xxxx/xxxxxxx/f0.mp4",
        "RoomId":311601250
    }
}
```


成员进入事件

事件类型

MemberJoin

事件说明

当成员进入房间时发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
UserId	String	用户 ID。
Role	Integer	角色。 枚举值： • 0：学生。

		• 1：老师。 • 2：助教。 • 3：巡课/督导。 示例值：0

示例

```
{
    "Timestamp":1679279225,
    "ExpireTime":1679279825,
    "Sign":"6fcaf48026fe9***************8ede",
    "SdkAppId":35***71,
    "EventType":"MemberJoin",
    "EventData":{
        "RoomId":366317280,
        "UserId":"2Lzh8d3R***************DiDn",
        "Role":0
    }
}
```

成员退出事件

事件类型

MemberQuit

事件说明

当成员退出房间时发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
UserId	String	用户 ID。
Reason	Integer	退出原因，详细见下表 Reason 枚举值。

Role	Integer	角色 枚举值： • 0：学生。 • 1：老师。 • 2：助教。 • 3：巡课/督导。 示例值：0

Reason 枚举值

值	描述
0	主动退出。
1	被踢。
2	永久被踢。
4	失去心跳下线。
5	房间结束，成员自动退出。

示例


```
{
    "Timestamp":1679279260,
    "ExpireTime":1679279860,
    "Sign":"03d41254d4ba7***************3d99",
    "SdkAppId":35***71,
    "EventType":"MemberQuit",
    "EventData":{
        "RoomId":397322814,
        "UserId":"2NG5xjpn***************1TPf",
        "Reason":0,
        "Role":1
    }
}
```


文档转码完成事件

事件类型

DocumentTranscodeFinish

事件说明

当文档转码完成（成功或失败）时发出通知。

数据字段

字段名	类型	描述
DocumentId	String	文档 ID。
State	Integer	文档状态。
Result	String	转码结果，如果成功则为转码后地址，如果失败则为错误码。
Info	String	转码信息。
Thumbnail	String	缩略图地址，PPT 缩略图一般有多页，第一页的地址为 thumbnail_url/1.jpg。

示例

```
{
    "Timestamp":1679281156,
    "ExpireTime":1679281756,
    "Sign":"1597c5c8aaafb***************e780",
    "SdkAppId":35***71,
    "EventType":"DocumentTranscodeFinish",
    "EventData":{
        "DocumentId":"sixkzoak",
        "Info":"",
        "Result":"https://xxx.cos.ap-
shanghai.myqcloud.com/doc/xxxxxx/picture/",
        "State":3,
        "Thumbnail":"https://xxxxx.cos.ap-
shanghai.myqcloud.com/doc/xxxxx/thumbnail/"
    }
}
```

文档创建事件

事件类型

DocumentCreate

事件说明

当客户端文档创建成功时发出通知。

数据字段

字段名	类型	描述
DocId	String	文档 ID。
DocName	String	文档名称。
Owner	String	文档拥有者的 UserId。
DocSize	Integer	文档大小 单位字节。
DocUrl	String	文档链接。
Permission	Integer	文档权限。0：私有文档；1：公共文档。

示例

```
{
    "Timestamp":1679281150,
    "ExpireTime":1679281750,
    "Sign":"44f0a2e422ede**************f51c",
    "SdkAppId":35***71,
    "EventType":"DocumentCreate",
    "EventData":{
        "DocId":"sixkzoak",
        "DocName":"test.pdf",
        "DocSize":4162606,
        "DocUrl":"https://xxxx.cos.ap-
shanghai.myqcloud.com/uploads/xxxxx/xxxxxx/xxxxxx.pdf",
        "Owner":"2Lzh8d3R**************DiDn",
        "Permission":0
```

```
    }
}
```

文档删除事件

事件类型

DocumentDelete

事件说明

当客户端文档删除成功时发出通知。

数据字段

字段名	类型	描述
DocId	String	文档 ID。

示例

```
{
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328***************3be9",
    "SdkAppId":35***71,
    "EventType":"DocumentDelete",
    "EventData":{
        "DocId":"sixkzoak"
    }
}
```

伪直播开始事件

事件类型

FakeLiveStart

事件说明

当伪直播开始时发出通知。

数据字段

字段名	类型	描述
class_id	Integer	房间 ID。

示例


```
{
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328***************3be9",
    "SdkAppId":35***71,
    "EventType":"FakeLiveStart",
    "EventData":{
        "class_id":324896216
    }
}
```


伪直播结束事件

事件类型

FakeLiveStop

事件说明

当伪直播结束时发出通知。

数据字段

字段名	类型	描述
class_id	Integer	房间 ID。
error.code	String	错误码（正常结束无此字段）。
error.message	String	错误描述（正常结束无此字段）。

示例


```
{
```

```
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328**************3be9",
    "SdkAppId":35***71,
    "EventType":"FakeLiveStop",
    "EventData":{
        "class_id":324896216,
        "error":{
            "code":"streamInterruption",
            "message":"streamInterruption"
        }
    }
}
```


自定义事件

事件类型

TaskUpdate

事件说明

在自定义 UI 中调用 JSAPI updateTask，并传入参数 `enableCallback = true` 时发出通知。

数据字段

字段名	类型	描述
RoomId	String	房间 ID。
TaskId	String	任务 ID（对应 updateTask 的 `taskId` 参数，可自定义）。
CustomData	String	自定义参数（对应 updateTask 的 `content` 参数）。

事件示例


```
{
    "Timestamp": 1679281184,
    "ExpireTime": 1679281784,
    "Sign": "964ff6d946328**************3be9",
    "SdkAppId": 35***71,
```

```
    "EventType": "TaskUpdate",
    "EventData": {
        "RoomId": "397322814",
        "TaskId": "your-task-id",
        "CustomData": "{\"key1\":\"value1\",\"key2\":\"value2\"}"
    }
}
```

JS 代码示例

```
TCIC.SDK.instance.updateTask(
  'your-task-id', // taskId
  JSON.stringify({ key1: 'value1', key2: 'value2' }), // content
  -1, // duration
  false, // createOnly
  '', // bindingUser
  true, // enableCallback
);
```

转存开始事件

事件类型

MixedFlowTransferStart

事件说明

当自定义转存开始时发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
TransferStatus.Code	String	错误码（TransferSucceed、TransferFailed、UnauthorizedOperation）。
TransferStatus.Message	String	错误描述（正常开始无此字段）。

示例

```
{
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328***************3be9",
    "SdkAppId":35***71,
    "EventType":"MixedFlowTransferStart",
    "EventData":{
        "RoomId":324896216,
        "TransferStatus":{
            "Code":"TransferSucceed",
            "Message":""
        }
    }
}
```

转存结束事件

事件类型

MixedFlowTransferEnd

事件说明

当自定义转存结束时发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
TransferDuration	Integer	转存文件时长，单位秒。
TransferSize	Integer	转存文件大小，单位 M。
TransferFileId	String	转存 VOD 文件 fileid。
TransferUrl	String	转存 VOD 的地址链接。
RecordUrl	Strin	互动课堂侧的录制文件地址。

	g	
TransferStatus.Code	String	错误码（0、TransferFailed、UnauthorizedOperation）。
TransferStatus.Message	String	错误描述。

示例


```
{
    "Timestamp": 1679281184,
    "ExpireTime": 1679281784,
    "Sign": "964ff6d946328***************3be9",
    "SdkAppId": 35***71,
    "EventType": "MixedFlowTransferEnd",
    "EventData": {
        "RoomId": 350389385,
        "TransferDuration": 8,
        "TransferSize": 6,
        "TransferFileId": "12***************14",
        "TransferUrl": "https://1500029853.vod-
qcloud.com/6cc0b3e3vodcq1500029853/7792888c1253642697459575214/f0.mp4",
        "RecordUrl":
"https://1500015970.vod2.myqcloud.com/6cac6e5evodcq1500015970/d6c3f4f41253
642697457297772/f0.mp4",
        "TransferStatus": {
            "Code": "0",
            "Message": "SUCCESS"
        }
    }
}
```


房间用量统计事件

事件类型

MemberStatistics

事件说明

当房间结束时发出通知。

数据字段

字段名	类型	描述
AudienceType	Integer	观看类型，互动观看（默认）。
ClassId	Integer	课程 ID，同房间 ID。
ClassName	String	课程名称，同房间名称。
Interaction	Integer	TRTC 总时长。
LowLatencyPresent	Integer	快直播总时长。
MaxRTCNumber	Integer	最大上麦人数。
MemberJoinNumber	Integer	进房总次数。
MemberNumber	Integer	累计在线人数。
RealEndTime	Integer	秒级 Unix 时间戳，实际房间结束时间。
RealStartTime	Integer	秒级 Unix 时间戳，实际房间开始时间。
Resolution	Integer	头像区域，摄像头视频画面的分辨率。可以有如下取值： • 1：标清 • 2：高清 • 3：全高清
SchoolId	Integer	应用 ID，AppId。

板书截图完成事件

事件类型

WhiteBoardSnapshotFinish

事件说明

当板书截图生成完毕后发出通知。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
Status	Integer	板书生成任务的状态。2：失败，3：成功。
Total	Integer	截图总数。
Result	Array of String	截图列表。

示例


```
{
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328***************3be9",
    "SdkAppId":35***71,
    "EventType":"WhiteBoardSnapshotFinish",
    "EventData":{
        "RoomId":324896216,
        "Result":[
          "https://xxx.cos.ap-
shanghai.myqcloud.com/snapshot****57037231948620.png",
          "https://xxx.cos.ap-
shanghai.myqcloud.com/snapshot****57037244238347.png"
        ],
        "Status":3,
        "Total":2
    }
}
```


成员开始推流事件

事件类型

PushStream

事件说明

成员开始推流时发出通知。

> **注意：**
> 目前仅支持直播类型为 “ RTMP 推流直播 ” 房间（ 创建房间 中 LiveType=2），身份为 “ 老师 ” 的用户开始推流事件的回调。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
LiveType	Integer	直播类型： • 0：常规（默认）。 • 1：伪直播。 • 2：RTMP 推流直播。
UserId	String	用户 ID。
Role	Integer	用户身份： • 0：学生。 • 1：老师。 • 2：助教。

示例

```
{
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328***************3be9",
    "SdkAppId":35***71,
    "EventType":"PushStream",
    "EventData":{
        "RoomId":324896216,
        "LiveType":2,
        "UserId":"2Lzh8d3R***************DiDn",
        "Role":1
    }
}
```

成员停止推流事件

事件类型

StopStream

事件说明

成员停止推流时发出通知。

> **注意：**
> 目前仅支持直播类型为“RTMP 推流直播”房间（ 创建房间 中 LiveType=2 ），身份为“老师”的用户停止推流事件的回调。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
LiveType	Integer	直播类型： • 0：常规（默认）。 • 1：伪直播。 • 2：RTMP 推流直播。
UserId	String	用户 ID。
Role	Integer	用户身份： • 0：学生。 • 1：老师。 • 2：助教。

示例


```
{
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328***************3be9",
    "SdkAppId":35***71,
    "EventType":"StopStream",
    "EventData":{
        "RoomId":324896216,
```

```
        "LiveType":2,
        "UserId":"2Lzh8d3R**************DiDn",
        "Role":1
    }
}
```

页面录制事件

事件类型

WebRecordFinish

事件说明

页面录制出现异常，导致录制自动结束时会发出通知。情况如下：

1. 录制时长达到设定的最大录制时长（24小时）。
2. 录制任务中出现超过最大分辨率限制（1920*1080）的视频或图片。

数据字段

字段名	类型	描述
RoomId	Integer	房间 ID。
EventType	Integer	状态码804，代表页面录制异常导致结束录制。
EventMessage	String	• Over time limit：录制时长达到设定的最大录制时长，自动结束录制。 • Over resolution limit：录制任务中出现超过最大分辨率限制的视频源，自动结束录制。

```
{
    "Timestamp":1679281184,
    "ExpireTime":1679281784,
    "Sign":"964ff6d946328**************3be9",
    "SdkAppId":35***71,
    "EventType":"WebRecordFinish",
    "EventData":{
        "RoomId": 345435412,
        "EventType": 804,
        "EventInfo": {
            "TaskId": "23453323432432",
            "Payload": {
```

```
            "Status": 1,
            "EventMessage": "Over time limit"
            }
    }
}
```

移动端原生内核与 Web 内核的区别

最近更新时间：2026-05-25 21:04:51

为了满足不同客户的业务场景、开发周期以及性能需求，实时互动-教育版的移动端 SDK 为您提供了两种不同的底层架构选择：**Web 内核与原生内核**。
本文档将介绍这两种内核的实现原理、优缺点及适用场景，帮助您快速选择最适合您项目的接入方案。

Web 内核

Web 内核是基于移动端 WebView 及高性能 Web 引擎技术实现的 SDK 方案。它通过在 App 内部加载网页视图来实现音视频互动与教室功能。

核心特点

- **技术栈**：开发者可以使用前端标准技术（HTML、CSS 和 JavaScript）进行业务逻辑的编写与自定义开发。
- **跨平台与高效率**：支持“一套代码，多端运行”（适配 iOS、Android 等平台），极大地降低了开发成本，适用于需要快速开发、敏捷迭代和迅速部署上线的场景。
- **性能与兼容性**：由于基于 Web 引擎渲染，其运行性能和操作流畅度相较于原生内核会有一定差距；同时，在面对市面上极其繁杂的移动设备机型时，可能无法做到100%的兼容。
- **授权限制（License）**：**每个旗舰版套餐内仅赠送一个 Web 引擎 License**。这意味着，如果您选择 Web 内核方案，一个旗舰版授权只能绑定并支持一个 App 包名（Package Name / Bundle ID）。

适用场景

- 项目周期紧张，需要快速上线验证业务。
- 团队以 Web 前端开发人员为主，缺乏移动端原生开发资源。
- 对极致性能要求不高，更看重跨平台开发效率的教育类应用。

原生内核

原生内核是基于移动端原生技术深度构建的 SDK 方案。它直接调用系统底层 API 进行音视频的采集、渲染与业务逻辑处理。

核心特点

- **卓越性能与体验**：原生内核在音视频处理延迟、画面渲染流畅度以及内存占用上具有显著优势，能够为用户提供极高稳定性和极佳的沉浸式上课体验。
- **功能丰富**：能够更深层次地调用设备硬件能力，提供比 Web 端更丰富、更底层的互动功能。
- **无包名限制**：原生内核**没有 Web 引擎的 License 绑定限制**，同一个授权可以支持多个不同的 App 包名，方便客户进行多产品矩阵的布局。
- **开发门槛**：如果客户需要对 UI 或业务逻辑进行深度的自定义开发，需要开发团队具备移动端原生开发能力，整体开发与测试周期相对较长。

适用场景

- 对音视频实时性、画面流畅度及应用稳定性有极高要求的场景。
- 拥有专业的移动端原生开发团队，追求极致用户体验。
- 旗下拥有多个 App 产品线（多个包名），需要共用底层互动能力的客户。

核心差异对比总结

为了便于您快速评估，我们对两种内核的核心差异进行了总结：

对比维度	Web 内核	原生内核
底层实现	基于 WebView 与高性能 Web 引擎	基于移动端原生技术构建
开发语言	HTML、CSS、JavaScript	原生开发语言
性能与稳定性	满足基础需求，性能逊于原生	**高性能、高稳定性**，体验好
跨平台能力	**极强**（一套代码适配多端）	需按原生规范进行集成与开发 提供 Flutter 跨平台 SDK
开发与部署周期	**短**（适合快速上线）	一般（适合精细打磨）
设备兼容性	良好（少数机器会出现初始化失败，而降级使用系统 WebView，存在不兼容问题）	**极佳**（深度适配系统底层）
包名限制 (License)	**有限制**（1个旗舰版仅支持1个包名）	**无限制**（支持多个包名）

选型建议

- 如果您的核心诉求是**“快”**——希望用最低的研发成本、最快的速度将线上教室集成到现有的多个平台中，且您的 App 只有一个主体包名，推荐选择 **Web 内核**。
- 如果您的核心诉求是**“稳”与“精”**——希望给师生提供媲美行业顶尖水平的流畅上课体验，或者您有多个 App 包名需要集成 SDK，推荐选择**原生内核**。

课堂录制指南

最近更新时间：2025-11-25 16:02:21

概述

本文将介绍如何快速使用实时互动-教育版云端录制功能，帮助客户在1对1互动教学、在线互动小班课、互动直播大班课等多种在线教学场景中，对课堂内容进行存档等多样化需求。目前，我们提供**页面录制**、**混流录制**和**手动录制**。

录制类型	说明
页面录制	指全场景录制页面内的全部元素，包括音视频画面，白板演示以及聊天窗口等各类内容，并保证通话内容与白板时间同步，可以完整地录制课堂（房间）的所有实时信息，即**用户视角看到的课堂页面信息**，所见即所得。 页面录制的特点是还原课堂情况，包括聊天、布局切换等。当前单次录制时长上限为**24小时**。
混流录制	指根据预先设定的录制模板，将多路音视频（如老师摄像头画面）和白板画面（如课件内容、共享屏幕画面）合成一个完整的视频，以便于记录课堂信息。 其特点在于能够根据**不同的录制模板生成相应的视频**，方便在各种场景中使用这些录制内容。 当前单次录制时长上限为**24小时**。
手动录制	指客户**按需手动启动/停止录制**，对应的业务场景更加灵活和定制。当前手动录制的模板样式同页面录制（即 `replay_layout = 9` ）方式。 如果一节课中会多次启动手动录制任务，并希望在课后可自动合成一个完整的录制文件。可在 创建课堂 时，通过 `RecordMerge = 1` 开启录制合并任务，即可在课后通过 获取课堂配置信息 的 RecordUrl 获取对应的录制文件（合并的录制文件时长上限为**24小时**）。

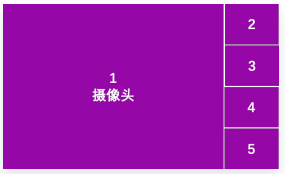
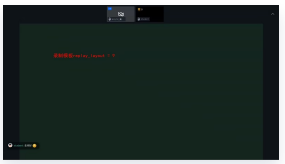
注意：

- 混流录制、手动录制，可通过 `DisableRecord` 进行设置。设置详情参见下文 配置步骤 。
- 如在音视频+白板课程中，开启页面录制，并使用 备用转码（TranscodeType=3）的课件，需要确保 UI 版本在1.9.8及以上。

选择录制模板

设置 `replay_layout` 指定录制模板，配置方式见配置 步骤1.2 。

参数设置	说明	模板	备注
replay_layout = 0	教师和学生区域（2、3、4、5所在区域）宽高比为1:1。 当源视频流比例不为1:1时则会裁剪显示。		如图所示，此类模板分为3个区域： • 白板/屏幕共享区域（1所在区域）。 • 老师区域（2所在区域）。

replay_layout = 3	教师和学生区域（2、3、4、5所在区域）宽高比为4:3。 当源视频流比例不为4:3时则会裁剪显示。		• 学生区域（3、4、5所在区域）。 **最多录制3路学生（或助教）的视频。根据上台顺序，确认录制学生（或助教）。**
replay_layout = 1	教师和学生显示区域（2、3、4、5所在区域）宽高比为1:1。 当源视频流比例不为1:1时则会裁剪显示。		如图所示，此类模板分为2个区域： • 老师区域（1所在区域）。 • 学生区域（2、3、4、5所在区域）。 **最多录制4路学生（或助教）的视频。** 当课堂类型为1V0的时候，只存在老师区域（1所在区域），并且会占据全屏。
replay_layout = 4	教师和学生显示区域（2、3、4、5所在区域）宽高比为4:3。 当源视频流比例不为4:3时则会裁剪显示。		
replay_layout = 5	教师显示区域(2所在区域)宽高比为1:1。 当源视频流比例不为1:1时则会裁剪显示。		如图所示，此类模板分为2个区域： • 白板/屏幕共享区域（1所在区域） • 教师显示区域（2所在区域）。 白板/屏幕共享区域：当有屏幕分享时，覆盖白板显示。
replay_layout = 2	教师显示区域(2所在区域)宽高比为4:3。 当源视频流比例不为4:3时则会裁剪显示。		
replay_layout = 9（页面录制）	/		用户视角看到的课堂页面信息。真实还原课堂情况，包括聊天、布局切换等。 当前页面录制仅支持Web、PC、iOS 和Android，暂不支持小程序。
replay_layout = 13	教师(1)和学生显示区域(2)宽高比为9:16。		该模板适用于移动端竖屏纯视频课堂模式。

			1为老师画面，当老师共享屏幕时，会覆盖老师摄像头画面。
replay_layout = 14	教师(1)显示区域宽高比为16:9，学生显示区域（2）宽高比为1:1。 当学生源视频流比例不为1:1时则会裁剪显示。		该模板适用于移动端横屏纯视频课堂模式。 • 当老师共享屏幕时，会覆盖老师摄像头画面。 • 当学生没有开摄像头时，学生显示区域会隐藏。

配置指引

注意事项

为了避免录制失败（失败后无法补救），请注意以下情况：

- 待录制的页面中，任何视频图片的分辨率**不能超过页面录制最大分辨率（1920 × 1080）**，否则将导致录制失败。
  - 上传 PPT 课件（内嵌视频或纯视频课件）时，云 API 会通过转码的方式以确保视频分辨率不超过页面录制的最大分辨率。
  - H5课件内嵌视频图片等内容时，由于无法获取相关内容信息，因此**在制作环节需确保视频分辨率不超过页面录制的最大分辨率**。
- 创建课堂的 `RecordScene` ，该字段为客户自定义场景参数。仅在 `recordlayout=9` （页面录制）的时候此参数有效。请确保正确配置各类参数以保证其有效性。
- 在使用伪直播功能时，若出现播放异常，建议您先对外链地址进行自查。具体自查步骤如下：

（1）检查视频是否卡顿（任选一种方式即可）

  - 使用 VLC 播放器：通过 VLC 播放该链接，观察播放过程中是否出现卡顿、缓冲或中断现象。单击查看 下载地址 。
  - 使用命令行工具：在终端中运行命令，通过 ffplay 播放视频流，直观判断播放流畅性。参考如下命令：

```
ffplay https://x.mp4
```

（2）检查网络传输性能

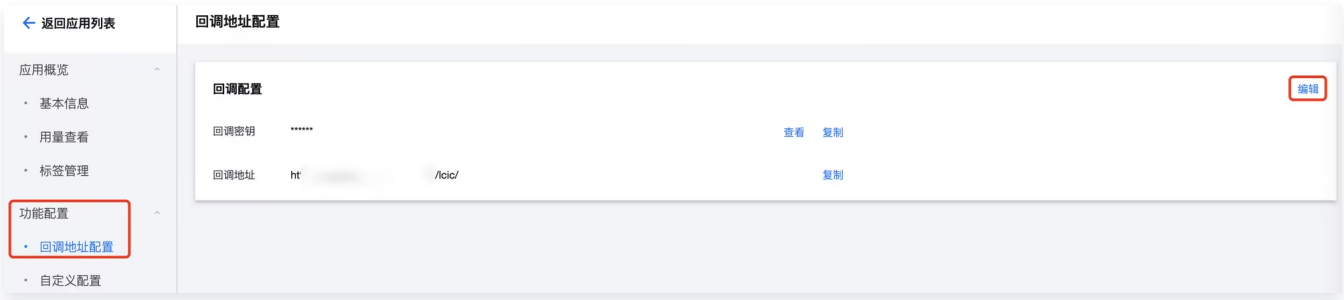
使用命令测试从服务器下载视频流的网络传输性能，重点关注传输速度和稳定性。参考如下命令：

```
time curl -o /dev/null https://x.mp4
```

配置步骤

1. **通过 创建房间 接口，设置录制信息。**
   1.1 通过 `DisableRecord` 设置录制方式。（枚举值：开启自动录制；禁止录制；开启手动录制）。
      - 录制将从课堂实际开始时启动，课堂实际结束时停止。可通过 获取房间统计信息，查询课堂实际开始和实际结束时间，参见 RealStartTime 和 RealEndTime。
      - 若采用**手动录制**，可通过 startRecord、stopRecord 接口控制录制的开始和结束。

   1.2 通过 `RecordLayout` 指定录制模板。课中不支持切换模板。
2. **前往 控制台 设置对应应用的回调地址。**
   操作界面如下图，具体操作参见 监听服务端事件回调。



3. **通过回调地址返回录制地址 RecordUrl。**
   - 该录制文件格式是.mp4。
   - 课堂结束后10分钟左右，系统将自动生成录制文件。若录制时长超过2小时，则录制文件生成会有延迟。

管理录制文件常见问题

如何存储录制文件?

实时互动-教育版会自动存储录制产生的文件，并产生存储费用，详见 计费概述。

如何删除录制文件?

实时互动-教育版支持单个或者批量删除录制文件，详情参考 录制文件管理相关接口。

如何找到投递到云点播的录制文件?

可通过服务端监听事件（MixedFlowTransferEnd）回调中的 TransferFileId、TransferUrl 拿到对应转存 VOD 的文件 fileId 和地址链接，详情参考 监听服务端事件回调。

信令录制

最近更新时间：2026-04-07 11:37:52

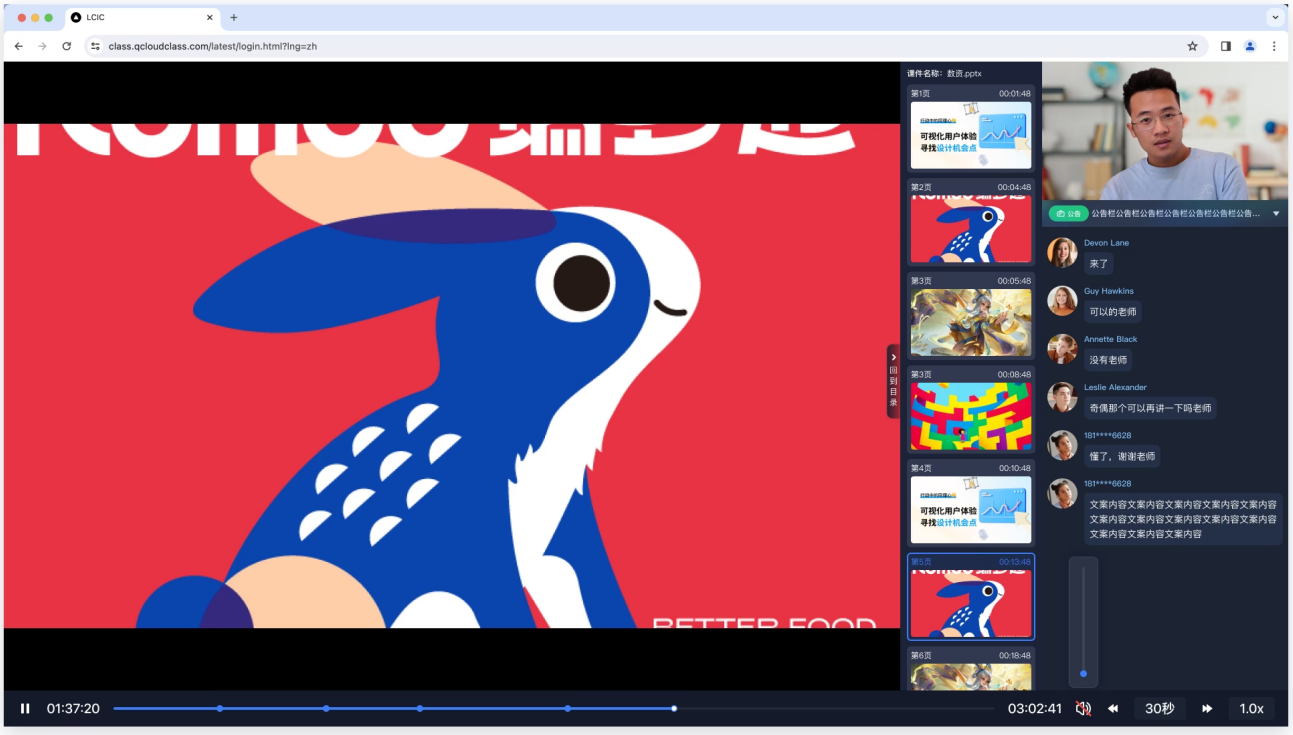
> **说明：**
> 本功能仅支持旗舰版用户。

功能介绍

信令录制功能可以将您的在线课程完整复刻为一个互动式回放。它能精准同步课堂中的每一个细节，包括：

- 教师的视频画面。
- 课件演示与白板板书。
- 聊天区的实时消息。

最终生成的回放不再是简单的视频文件，而是一个高度还原的在线课堂。学生可以像参与直播一样，获得身临其境的回放体验，不错过任何知识点和互动细节。



使用流程

步骤 1： 创建课堂

使用接口 CreateRoom 创建课堂时，重点关注以下参数配置，其余参数参考 接口文档 即可。

参数	说明

DisableRecord	开启信令录制填写 3。
RecordLayout	信令录制时，录制模板不生效，无需设置该参数。
RecordStream	信令录制时，无需设置该参数。

步骤 2： 观看信令录制

2.1 通过接口拿到 Token

通过 GetPlaybackToken 或 BatchGetPlaybackToken 接口拿到 Token。

2.2 打开录制页面

URL 示例：

```
https://class.qcloudclass.com/playback/index.html?classid={classid}&token={token}
```

参数说明：

参数名	是否必填	说明
classid	必填	课堂 ID。
token	必填	录制 Token。 通过 GetPlaybackToken 或 BatchGetPlaybackToken 接口获取。
title	选填	自定义显示页面的标题。
report	选填	是否上报学生观看的相关统计。 • false：默认值，不上报。 • true：开启上报。

其他相关接口

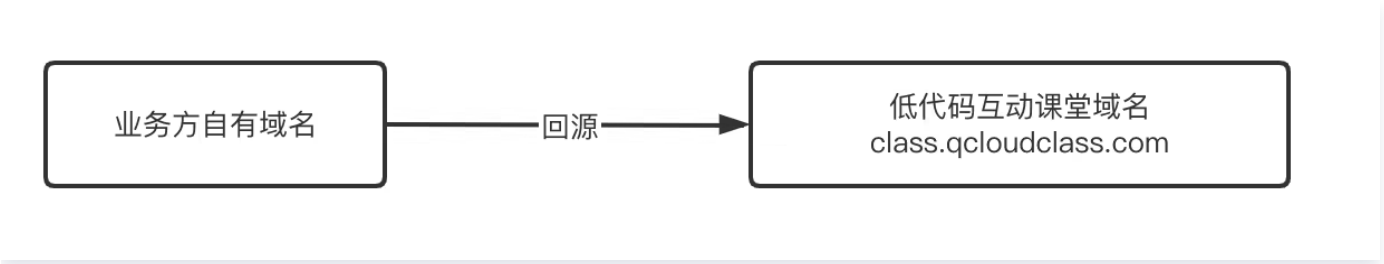
包括删除、查询等接口，可以参考 信令录制回放相关接口 。

自定义业务域名

最近更新时间：2026-05-09 22:01:51

想使用自有域名完成实时互动-教育版 Web 端的访问：
从地址 `https://yourdomain.com/` 转为我们提供的课堂实际域名.

方案



回源域名获取

我们提供两组域名，供不同位置的用户使用。请根据用户实际情况，选用合适的域名。

域名	服务器地域	说明
class.qcloudclass.com	中国大陆（全球加速）	中国大陆用户及全球不确定位置用户选用。
www.tencentclass.com	新加坡（全球加速）	非中国大陆用户首选访问，海外主体独立域名且纯海外原生部署，性能优化，全球可用性高。

操作步骤

> **说明：**
> 以 腾讯云 CDN（内容分发网络）为例。
> 我们假设业务层拥有一个课堂专属三级域名： `class.yourdomain.com`

一. 配置 CDN

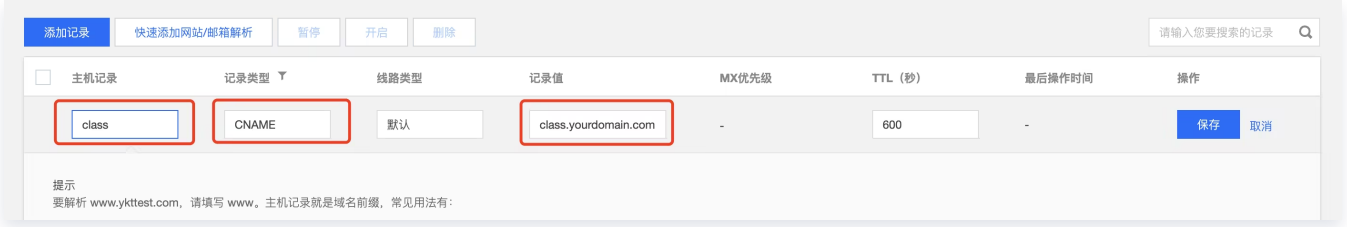
1. 前往 内容分发网络 控制台，单击**添加域名**进行添加。

2. 配置回源参数。填写相关域名配置和源站配置。源站配置请参考 对应的地域选择合适的域名，请选择主域名.

3. 添加成功后，单击域名查看基础配置信息，复制 CNAME。

二. 配置域名解析

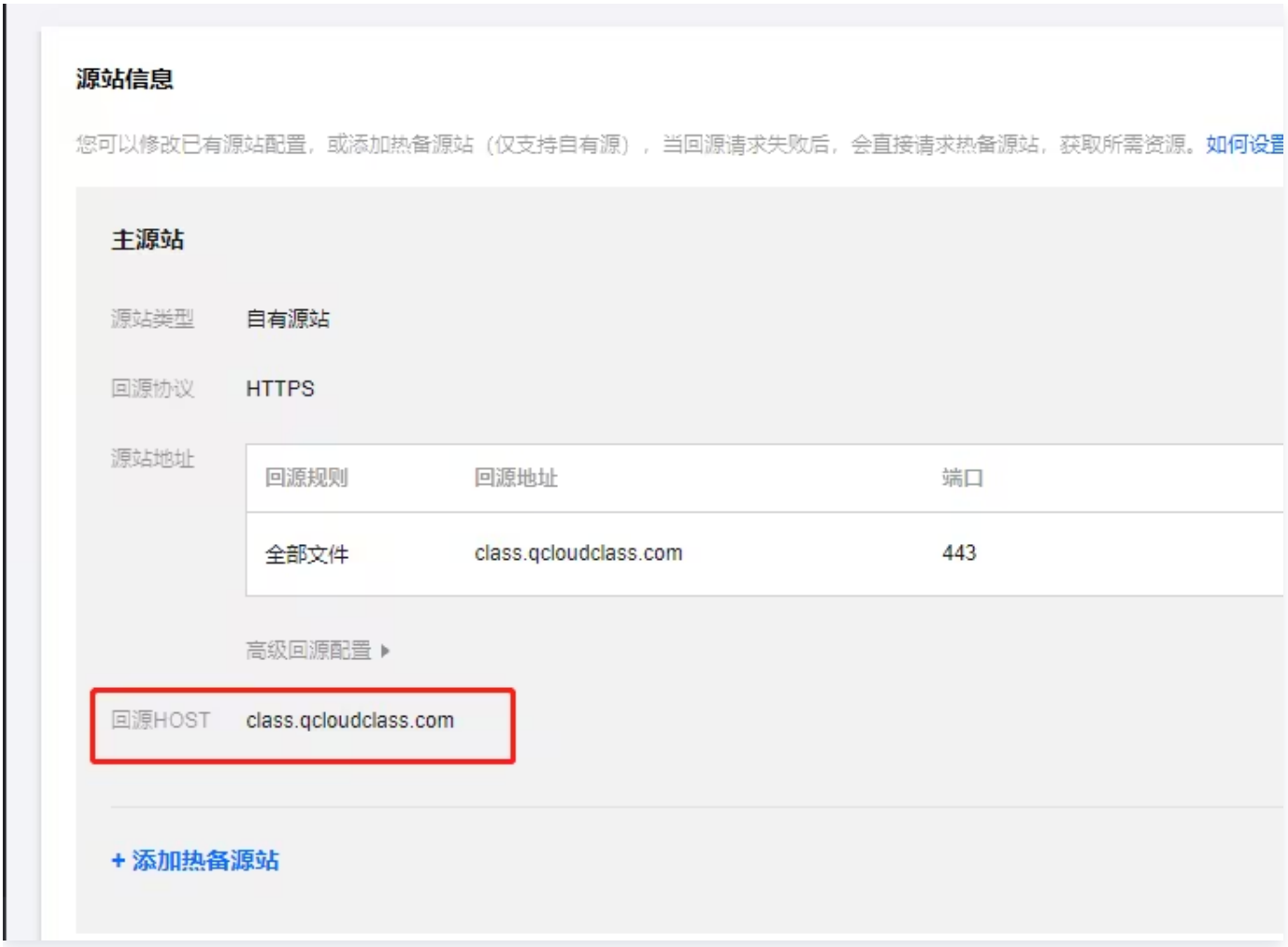
1. 前往域名解析配置 class.yourdomain.com 的域名指向。
2. 配置域名解析指向，以 云解析 DNS 为例。类型选择 CNAME，记录值为前面步骤获取的 CNAME 值。



3. 单击**保存**。

三. 配置回源 HOST

进入**回源配置**页面，修改 回源 `HOST` => 课堂源站域名 。回源 HOST 域名配置请参考 对应的地域选择合适的域名，请选择主域名.



四. 进入课堂

当您实际使用自定义域名进入课堂时, 请在 URL 中参数拼接 region 字段, 可选值 `cn` 或 `sg` . 对应请求中国大陆源站或新加坡源站, 显式声明用于进一步整体优化访问质量.

如:

`https://${自定义域名}/latest/class.html?classid=319068870&region=sg&其他更多参数...`

优化配置

为了让自定义业务域名能达到更好的访问效果，还需要进行以下调整。

一. 完成 https 配置

在 域名管理 页面，单击域名后选择 HTTPS 配置去进行配置。

二. 完成节点缓存过期配置

在缓存配置页面进行节点缓存过期配置。

> **说明：**
> 无需对 html 文件进行缓存。

三. 完成 gzip 配置

在高级配置页面，配置智能压缩规则。

四. 缓存键规则配置

在**缓存配置**页面，配置缓存键规则。

资源刷新

请参见 CDN 刷新预热 。

> **为何需要做资源刷新?**
> 根据实际的运营需求/Bug 修复/体验优化，实时互动-教育版会对线上版本不定期进行热更新。
> CDN 的缓存策略决定可能出现业务自有域名与实时互动-教育版官方域名表现不一致的情况。
> 我们建议业务侧定期进行资源的刷新操作，或按需进行立即刷新。


腾讯云
总览
云产品
前端性能监控
对象存储
互动白板
即时通信 IM
内容分发网络 CDN
内容分发网络
服务概览
域名管理
证书管理
统计分析
刷新预热
日志服务
插件中心
安全防护
服务查询
全网状态监控
流量包管理
IP归属查询
回源节点查询
自助诊断工具
内容合规
配额管理
解决方案
云开发
体验调研
刷新预热
URL刷新
目录刷新
URL预热
操作记录
• 当您的源站有资源更新/需要清理违规资源/域名有配置变更，为避免全网用户受节点缓存影响仍访问到旧的资源/受旧配置的
• 当您有运营活动或安装包/升级包发布等，可提交预热任务，提前将静态资源预热至 CDN 加速节点，降低源站压力，提升用
• 当您的业务活动、运营场景需要临时增加刷新或预热配额时，可以通过配额管理申请调整所需时间范围的临时配额。详细说
URL
输入内容
上传文件
https://class.yourdomain.com/
目录 URL 必须包含 http:// 或 https://，例如 http://www.test.com/test/，一行一个
中国境内 - 今日剩余个数：100
中国境外 - 今日剩余个数：100
选择刷新方式
刷新变更资源
刷新全部资源
URL Encode
开启后，将自动对带有特殊字符的url进行urlencode。
提交并刷新
刷新任务生效时间大约需要5分钟
查看结果请前往 操作记录

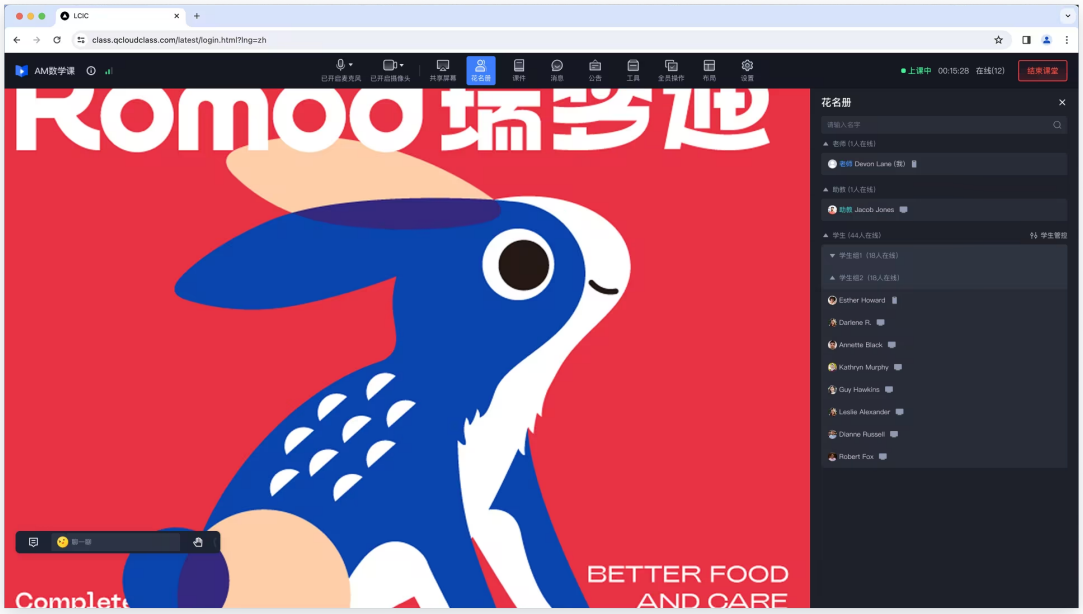
分组直播

最近更新时间：2026-02-13 17:14:52

> **说明：**
> 本功能需要开通白名单才可使用，仅支持旗舰版。您可通过 联系我们 里的微信群进行申请。

功能介绍

分组直播功能允许您将一个主课堂拆分为多个并行进行的子课堂。所有子课堂共享主讲老师的音视频画面和教学内容，但各子课堂内的学生之间相互隔离，无法互通。
老师视角可以看到所有子课堂：



> **注意：**
> 分组直播使用限制：
> 1. 需开通**旗舰版**方可使用。
> 2. 最多支持创建10个分组。
> 3. 仅支持1名学生与老师进行全局连麦。

在分组直播模式下，不同角色的权限范围如下。为了方便理解，我们定义了两种可见范围：

- 全局可见：该内容对所有分组的师生可见。
- 组内可见：该内容仅对所在分组的师生可见。

角色\功能	视频画面	白板/共享屏幕	互动消息	教学工具	花名册

老师	全局可见	全局可见	全局可见	全局可见	老师管理所有分组
助教	不适用	不适用	组内可见	不适用	仅管理所在分组
连麦学生	全局可见	全局可见	组内可见	不适用	不适用
未连麦学生	不适用	不适用	组内可见	不适用	不适用

开发流程

第一步：创建课堂

使用接口 CreateRoom 创建课堂时，重点关注以下参数配置，其余参数参考接口文档即可。

参数	说明
RoomType	填4代表分组直播。
MaxMicNumber	最大连麦人数，使用分组直播时只能填0或1。

第二步：创建分组

使用接口 CreateGroupLiveCodes 创建分组，生成**分组直播参加码**。

第三步：进入指定分组

学生和助教进入课堂时，需要指定其分组。（老师无需额外操作）

1. 获取分组码

使用服务端接口 DescribeGroupLiveCodes 获取分组码。

2. 进入分组课堂

Web 端

学生或助教进房时需要在课堂 URL 上拼接分组码，在原 URL 上拼接 `groupLiveCode={分组码}`，示例：

> **注意：**
> - 分组码包含#号，因此拼接 URL 时，必须进行编码：encodeURIComponent(groupLiveCode) 。
> - 老师的 URL 不需要拼接 groupLiveCode 参数。

```
https://class.qcloudclass.com/latest/class.html?
userid=${userid}&token=${token}&classid=${classid}&schoolid=${schoolid
}&groupLiveCode={groupLiveCode}
```

electron 客户端

在 `customParams` 里传参 `groupLiveCode`，示例代码：

```
const TCIC = require('tcic-electron-sdk')
TCIC.initialize({
  classId: '368507569',
  userId: '123456',
  token: 'token',
  // 如果提供了url， 则url参数优先级高于上述参数
  url: 'https://class.qcloudclass.com/latest/class.html?
classid=xxxx&userid=xxx',
  customParams?: {
    groupLiveCode: '@TGS#_368507569@TOPIC#_d5ou4vg9g8h8vaskt13g' // 分
组直播学生端需要传入
  },
  //其余参数请参考electron文档
})
```

Android(Web 内核)

Android 需要 1.8.26 及以上版本才支持，示例代码：

```
Intent intent = new Intent(getActivity(), TCICClassActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
Bundle bundle = new Bundle();
TCICClassConfig initConfig = new TCICClassConfig.Builder()
    .schoolId(schoolId)
    .classId(classId)
    .userId(userId)
    .token(token)
```

```
    .groupLiveCode(groupLiveCode)     //分组直播
    .build();
bundle.putParcelable(TCICConstants.KEY_INIT_CONFIG, initConfig);
intent.putExtras(bundle);
startActivity(intent);
```

iOS(Web 内核)

iOS 需要1.8.5.18及以上版本才支持，示例代码：

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];
roomConfig.schoolId = 123456;
roomConfig.userId = "test";
roomConfig.token = "test_token";
roomConfig.classId = 654321;
roomConfig.groupLiveCode =
"@TGS#_368829238@TOPIC#_d5nf8ud2et7qpiel1260";  //分组直播
[roomConfig setValue:@"en" forKey:@"language"]; //语言设置，可选
[roomConfig setValue:@"scene_name" forKey:@"scene"]; //可根据场景配置不同
的定制，可选
[roomConfig setValue:@(0) forKey:@"preferPortrait"]; //默认横屏，可选（0
是横屏，1是竖屏）

TCICClassController *vc = [TCICClassController
classRoomWithConfig:roomConfig];
if (vc) {
    [(UINavigationController *)self.window.rootViewController
pushViewController:vc animated:YES];
}else {
    NSLog(@"参数有误");
}
```

原生内核

正在开发中，敬请期待。

圆桌会议

最近更新时间：2026-03-24 16:59:52

> **说明：**
> 本功能需要开通白名单才可使用。您可通过加入 联系我们 里的微信群进行联系并申请。

功能介绍

圆桌会议是一种专为大型研讨、高端论坛等场景设计的互动课堂模式。该模式支持**1名主讲老师与最多6名嘉宾**同时上台进行音视频连麦互动，为用户提供沉浸式的实时沟通体验。

> **注意：**
> 需要购买旗舰版，并且开通后付费才能使用圆桌会议功能。

开发流程

第一步：创建课堂

业务后台需调用 CreateRoom 接口来创建圆桌会议房间。除常规参数外，请重点关注并按如下要求配置特定参数：

参数名	类型	描述与配置要求
RoomType	Integer	**房间类型**。 固定填 `3`，代表圆桌会议场景。
MaxMicNumber	Integer	**最大连麦人数**。 取值范围为 `0` ~ `6`（圆桌会议最多支持 6 人同时连麦）。
DisableRecord	Integer	**禁用录制状态**。 固定填 `0`。圆桌会议场景下**必须开启**自动录制。
RecordLayout	Integer	**录制布局**。 圆桌会议场景下，仅支持传入 `16` 或 `17`。
Guests.N	Array of String	**嘉宾 ID 列表**。 用于提前指定受邀嘉宾，示例值：`["guest_id_01", "guest_id_02"]`。

> **说明：**
> 其他未列出的参数，请直接参考 CreateRoom 接口官方文档的默认说明进行配置。

第二步：开始上课

课堂创建成功且师生进入房间后，可以通过以下两种任一方式开始上课：

- **客户端操作**：主讲老师进入课堂客户端后，手动点击界面右上角的“上课”按钮。
- **服务端调用**：由业务后台直接调用 StartRoom 接口，通过信令控制课堂开始。

界面效果示例

1. 老师界面

老师端拥有课堂控制权，可管理上台嘉宾及课堂进程。效果参考如下：

2. 学生界面

效果参考如下：


vik测试圆桌2
微信咨询
该平台专注提供线上
上课中：00:14:25(1人在线)
消息
公告
离开
什么是低代码互动课堂?
基础PaaS一站式集成
实时音视频 TRTC
IM
快直播
标准直播
点播
CDN
AI降噪
虚拟背景
场景化功能按需接入
房间管理
用户&麦位管理
互动白板&课件
屏幕共享&录制回放
标准化+自定义UI
LCIC demo
自有品牌的低代码互动课堂
1v1 在线辅导
互动小班课
万人直播公开课
伪直播大班课
拍摄直播课
双师课堂
AI互动课堂
研讨会
低门槛快速接入，灵活定制UI
节省90% 开发时间，15分钟上线自有品牌的互动课堂，支持自定义UI和业务自有插件，满足多样化场景
丰富教学工具，体验升级
结合上麦互动、互动白板、教学课件、屏幕共享、消息互动、云端录制、计时答题等，满足线上教学多样化需求
全球网络覆盖，保障用户体验
实时传输和直播分发网络，覆盖200个国家地区。在高丢包率高抖动的弱网环境下，仍保证音视频互动体验
全终端互通
提供Web/H5 、Windows 、MacOS 、Android 、iOS、小程序全终端、全平台接入方案
聊一聊

伪直播和 RTMP 推流课堂

最近更新时间：2026-03-27 14:28:22

功能介绍

伪直播课堂支持将已有的录播视频或 RTMP 视频流作为直播源进行推流上课，为学生提供与真实直播一致的沉浸式学习体验。

> **说明：**
> 伪直播功能仅限标准版及以上版本的客户使用。

开发流程

第一步：创建课堂

业务后台需调用 CreateRoom 接口来创建伪直播课堂房间。除常规参数外，请重点关注并按如下要求配置特定参数：

参数名	类型	描述与配置要求
`LiveType`	Integer	直播类型。取值： • `1`：伪直播（录播视频推流）。 • `2`：RTMP 推流直播。
`RecordLiveUrl`	String	视频源 URL。 • 当 `LiveType=1` 时，需在此填入录播视频的 URL。 • 当 `LiveType=2` 时，无需填写此参数。课堂创建后，可通过调用 DescribeRoom 接口获取本课堂的 RTMP 推流地址。
`SubType`	String	课堂子类型。取值： • `videodoc`：文档 + 视频 • `video`：纯视频 **建议：**推荐使用 `video`，此时视频流将直接显示在主画面区域，观看效果更佳。
`MaxMicNumber`	Integer	最大连麦人数。 **建议：**推荐填 `0`，将整个视频流展示在页面中央；您也可以根据实际业务需求进行调整。

> **说明：**
> 其他未列出的参数，请直接参考 `CreateRoom` 接口官方文档的默认说明进行配置。

第二步：开始上课

课堂创建成功且师生进入房间后，可以通过以下两种方式之一开始上课：

1. **客户端操作**：由于伪直播课堂无需真实老师授课（老师角色无法进入），可安排“助教”进入课堂，手动点击界面右上角的“上课”按钮开启课程。
2. **服务端调用（推荐）**：由业务后台直接调用 StartRoom 接口，通过服务端信令控制课堂开始。

多平台同步直播（RTMP 转推）

最近更新时间：2026-03-27 14:28:22

功能介绍

实时互动-教育版支持将课堂中的音视频直播流，通过 RTMP 协议同步转推至第三方直播平台（如微信视频号、抖音、B站、快手等）。
通过该功能，教育机构可以打破单一平台限制，实现课程内容的多渠道分发，有效拓宽公域引流渠道，大幅提升课程曝光度与品牌影响力。

> **注意：**
> 需要购买旗舰版，并且开通后付费才能使用 RTMP 转推功能。

费用说明

使用 RTMP 转推功能，除了产生基础的音视频上课时长费用外，还会产生以下增值服务费用：

1. **RTMP 转推费用**：按实际转推的分钟数进行后付费结算。
2. **混流转码费用**：仅当您选择混流方式（即 `RelayType=1` ）时收取，按实际混流处理的分钟数计费。（目前转推不支持自定义分辨率，默认使用全高清进行转推）

> **说明：**
> 具体的计费规则与单价，请参考官方文档 计费概述 。

开发流程

第一步：创建课堂并开启转推

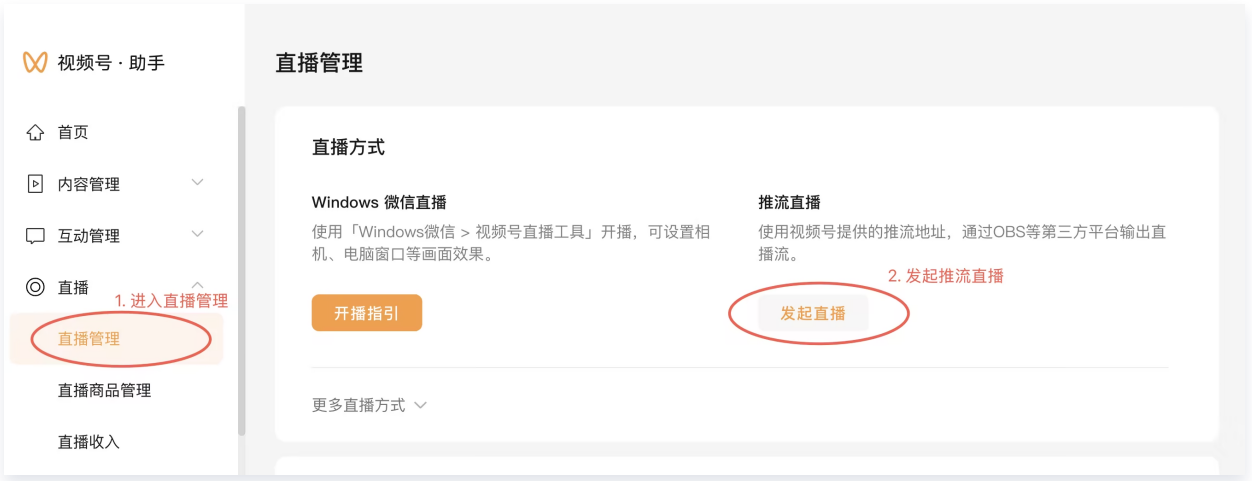
在创建课堂时，需明确开启转推权限。调用 CreateRoom 接口时，请重点关注并配置以下参数（其余参数按接口文档常规配置即可）：

参数名	类型	描述
EnableLiveRelay	Integer	**转推开关**。 填 `1` 代表开启转推；不填或填 `0` 代表关闭。

第二步：获取第三方平台推流地址（以微信视频号为例）

在配置转推前，您需要先在第三方直播平台获取推流地址（RTMP URL）。下文以微信视频号为例演示获取拼接过程：

1. 登录 视频号助手 ，进入左侧菜单的 **直播管理**，单击**推流直播**下方的**发起直播**。



2. 在推流信息页面，复制**推流地址**和**推流密钥**。

3. **拼接推流 URL**：将“推流地址”和“推流密钥”直接拼接在一起，即为下一步接口所需的完整转推 URL。示例如下：

```
// 拼接后的完整 URL 格式示例
"rtmp://1xx5x3.livepush.myqcloud.com/trtc_1400xxxxxx/live_2078923101xxxxxx?txSecret=178xxxe23d9xxxxxxxxf9xxba9f&txTime=xxxxxx"
```

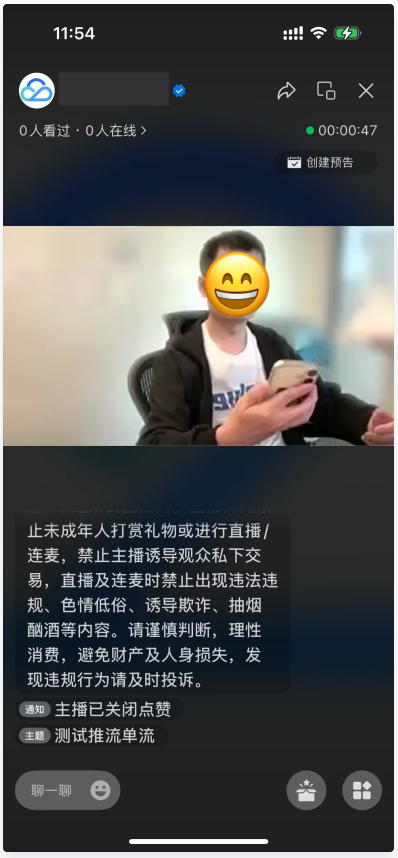
第三步：配置转推 URL 与画面布局

获取到推流 URL 后，调用 ModifyLiveRelayConfig 接口将画面推送到第三方平台。请重点关注以下参数：

参数名	类型	描述
Urls.N	Array of String	**转推目标地址**。 填写上一步拼接好的完整转推 URL，最多支持同时转推到10个平台。
RelayType	Integer	**转推画面类型**，支持 `0` 和 `1` 。 • `0` (单流)：仅推送老师的摄像头视频流。 • `1` (混流)：推送老师摄像头与白板拼接后的混合画面。

第四步：开始上课与效果展示

转推配置完成后，当课堂状态变为“上课中”时，系统会自动将直播流转推至第三方平台。
触发方式：老师进入课堂客户端后点击右上角的“上课”按钮，或由业务后台调用 StartRoom 接口开始上课。
不同 `RelayType` 在微信视频号的观看效果对比如下：

`RelayType=0` （单流效果）	`RelayType=1` （混流效果）
	

独立版设备检测-课堂小助手

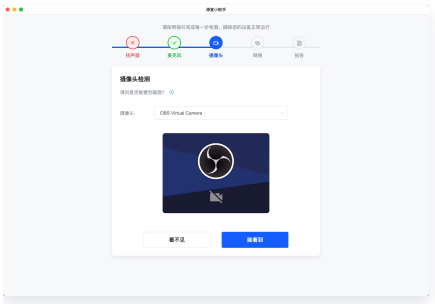
最近更新时间：2026-04-13 14:36:11

功能介绍

为帮助教师、学生等课堂参与者在上课前完成设备调试，我们提供独立的课前检测功能页面，与课堂主页分离。您可以单击此处 **立即体验** 。
支持自定义配置，检测以下项目：

- 扬声器
- 麦克风
- 摄像头
- 网络（系统网络环境及课堂使用的 HTTP 请求和 WSS 域名）
- 系统环境（TRTC、IM 等）

提供可视化检测报告汇总，并支持导出完整检测报告。
效果如下：



用户可按照进度条完成各项检测（可自定义开启检测项目）。完成后，检测结果将自动应用于课堂中。系统会默认选择已通过检测并修复的设备配置及域名配置，确保用户获得最佳初始体验。

集成指引

我们前期提供直接访问 URL、iframe 集成、Electron SDK、移动端 Mobile Native SDK 集成等多种接入方式供您选择。
后期将提供 npm 包，方便您在 Web 项目中直接使用。

URL 及参数

该部分 URL 及参数适用于下方所有接入方案，可统一参考。目前无论采用何种接入方式，均需使用该 URL (移动端 Native SDK 仅需使用这里的参数)。
线上设备检测链接： 单击此处进行检测

参数	默认值	说明
list	1111	四位数字位掩码控制，每位1代表启用，0代表不启用。顺序分别是：

		第一位： 扬声器 第二位： 麦克风 第三位： 摄像头 第四位： 网络及系统环境 例如：0101代表只启用麦克风和网络及系统环境检测。
language	auto	语言，目前支持简体中文和英语，具体 value 为： • auto：语言自适应 • zh：简体中文 • en：英文
theme	light	主题配色 • light：亮色模式 • dark：暗色模式

Web

Web 提供两种方式供用户使用，直接访问上述 URL 及通过 iframe 嵌入您的网页中。

访问 URL

用户可直接访问 线上地址链接 进行设备检测。
检测完成后，结果数据将自动存储在 localStorage 的 `tic_last_check_result` 字段中。课堂页面（1.10.3版本后）在加载时会自动读取该配置，作为网络域名及设备的默认参数。
若您的课堂项目使用自定义域名，请确保将设备检测页面也通过 CDN 回源至同一域名，以保证设备检测与课堂运行在相同域名下。

iframe 集成

您可以将该功能通过 iframe 集成到任意页面中。只需将 iframe 的 src 属性设置为 上述设备检测链接地址 即可。
检测完成后，设备检测会通过 postMessage 向外层发送事件，其中包含检测结果。格式如下：

```
{
type: 'TIC_DEVICE_CHECK_RESULT',
payload: result
}
```

请记录该结果，并在用户进入课堂时，将获取到的 result 对象原样拼接到课堂进房链接中，例如：
&tic_last_check_result={}
如果课堂页面同时在 URL 中指定了检测结果，且 localStorage 中也有同域结果，则优先采用 URL 中指定的结果。

Electron

Electron 端目前提供 SDK 接入的方式。

1. 由于部分依赖库的原因，请确保您使用以下基础库版本。

开发框架	版本
Node	16.14.2

2. 在您的项目中使用 npm 命令安装 SDK 包。

```
npm install tcic-electron-sdk@latest
```

> **说明：**
> - TCIC Electron SDK 最新版可在 tcic-electron-sdk 中查看。
> - **需要确保版本大于 1.10.1。**

3. 在项目脚本里引入模块后，调用 `openSiteWithUrl` 接口并传入之前获取的 URL 调起设备检测页面。

```
const TCIC = require('tcic-electron-sdk');

TCIC.openSiteWithUrl({
    url: `https://class.qcloudclass.com/devicecheck/index.html?
list=1111`, // list 请必传指定 electron 内
    onReady: (mainWindow) => {
      console.log('TCIC Device Check ready', mainWindow);
    },
    onClose: () => {
      console.log('TCIC Device Check  close');
    }
  });
```

4. 设备检测完成后，结果会记录在 Electron 的 localStorage 内，同域名下，并自动应用于课堂进房中。
   您后续调用 `TCIC.initialize` 方法进入课堂，就会自动应用设备检测结果。

移动端 Mobile Native

Mobile Native 提供 SDK 接入的方式。使用和课堂主应用同一个 SDK。

iOS

1. 安装并配置好腾讯云实时互动-教育版 iOS 端 Native SDK（可参考此文档）。**SDK 版本号要求: 最低 1.8.5.19。**
2. 根据上述文档步骤5指引，开发相同的课堂进房流程，调起实际页面。

与之不同的是，如需进入设备检测页，无需配置该文档中的 schoolId、classId、userId、token 等现有所有参数，请配置以下参数。

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];
NSMutableDictionary<NSString *, NSString *> *customParams =
[NSMutableDictionary dictionary];
customParams[@"deviceCheck"] = @"true"; // 配置为 true 则进入设备检测
customParams[@"deviceCheckParams"] = @"{\"list\": 1111}"; // 这里的具体参
数，请参考本文档 “URL 及参数” 章节
roomConfig.customParams = [customParams copy];
```

3. 设备检测完成后，结果会记录在 iOS SDK 的 localStorage 内，同域名下，并自动应用于课堂进房中。
   您后续正常进入课堂，就会自动应用设备检测结果。

Android

1. 安装并配置好腾讯云实时互动-教育版 Android 端 Native SDK（可参考此文档）。**SDK 版本号要求: 最低 1.8.27**。
2. 根据上述文档步骤4、步骤5指引，申请 SDK 授权，并初始化 Web 引擎。请确保 Web 引擎初始化完成，才可执行后续操作。
3. 根据上述文档步骤7指引，开发相同的课堂进房流程，调起实际页面。
   与之不同的是，如需进入设备检测页，无需配置该文档中的 schoolId、classId、userId、token 等现有所有参数，请配置以下参数。

```
Map<String, String> customParamsMap = new HashMap<>();
customParamsMap.put("deviceCheck", "true"); // 配置为 true 则进入设备检测
customParamsMap.put("deviceCheckParams", "{\"list\": 1111}"); // 这里的具
体参数，请参考本文档 “URL 及参数” 章节

TCICClassConfig.Builder builder = new TCICClassConfig.Builder()
        .customParams(customParamsMap);
```

4. 设备检测完成后，结果会记录在 Android SDK 的 localStorage 内，同域名下，并自动应用于课堂进房中。
   您后续正常进入课堂，就会自动应用设备检测结果。

结果导出

除了自动将结果应用于课堂项目外，用户还可以手动点击结果页面的"**复制结果**"，导出完整的检测流程及检测记录。
如需我们协助排查用户设备问题，请复制并发送完整的检测结果给我们。

定制化开发

如果需要对界面内容、交互进行调整，我们提供设备检测 SDK 及 UI 组件 npm 包供您自行接入并部署使用。其中：

- 无 UI SDK：@tencent-classroom/device-check-sdk 包括课堂所用的麦克风、摄像头、扬声器、网络及系统环境检测所有底层能力，开箱即用。开发产物为 Web 项目，需自行部署线上 URL，支持 Web、Electron (需配合上述的课堂 SDK)及 Mobile Native (需配合上述的课堂 SDK) 使用。具体用法可以参见 README 说明。
- UI：@tencent-classroom/device-check-ui 基于上方 SDK 的一套设备检测 UI，初始自带的默认界面和交互与提供的上述 免开发线上设备检测连接 相同。您可创建 Web 项目引入使用，并通过参数做一些基础的定制，或直接基于其 源码 GitHub 仓库 本地引入后修改界面及交互源码来使用。具体用法可以参见 README 说明。

无论您采用何种等级的定制化二次开发，基于 npm 包亦或是直接改代码，您均需要发布编译后的页面至线上。建议发布的设备检测页面和课堂同域同源，因设备检测完成后，会自动把检测结果选优数据(媒体设备、域名)配置到 localStorage 中，课堂主项目启动时，会从这里面引入配置，优化上课体验。
为了在项目中实际加载到您的定制版设备检测页面，还需要做如下的配置。

Web

用户直接访问定制的设备检测 URL 即可，无额外配置。

Electron

完成上述 Electron SDK 的引入后，实际启动设备检测窗口，请在调用 `TCIC.openSiteWithUrl` 方法时， `url` 传入定制的设备检测页面 URL。具体的参数配置可继续保持用我们内置的，也可以在 UI 层按需调整。

```
const TCIC = require('tcic-electron-sdk');

TCIC.openSiteWithUrl({
    url: `https://${您定制的设备检测页面URL}?list=1111`, // list 请必传指定
electron 内
    onReady: (mainWindow) => {
      console.log('TCIC Device Check ready', mainWindow);
    },
    onClose: () => {
      console.log('TCIC Device Check  close');
    }
  });
```

Mobile Native 移动端

请参照上述通过接入课堂移动端 SDK 以进入移动端设备检测的方案，在配置 TCICClassConfig 时，额外指定 `deviceCheckUrl` 参数，传入不含参数的 URL，即可进入定制的设备检测页面。
参数请通过 `deviceCheckParams` 方式指定，会拼接在 URL 上。具体的配置参数，可继续使用我们内置的，也可以在 UI 层按需调整。

移动端其他接入步骤请参见前述章节.

Android

```
Map<String, String> customParamsMap = new HashMap<>();
customParamsMap.put("deviceCheckUrl", "定制的设备检测链接")
// ... 其他用法保持不变
```

iOS

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];
customParams[@"deviceCheckUrl"] = @"定制的设备检测链接";
// ... 其他用法保持不变
```

联系我们

您有任何问题，可以扫码添加微信群与我们联系。

移动端 APP 增加屏幕共享（Web 内核）

最近更新时间：2023-10-07 16:44:02

iOS 端

1. 创建 `App Group`，参见实时音视频文档中 步骤 1：创建 App Group 。
2. 创建 `Broadcast Upload Extension`，参见实时音视频文档中 步骤 2：创建 Broadcast Upload Extension 。
3. 为新创建的 `Target`，依赖 `TCICSDK_ReplayKit`，如下代码，之后重新 `pod install` 即可。

```
target '新target名' do
# Comment the next line if you don't want to use dynamic frameworks
# use_frameworks!
pod 'TCICSDK_ReplayKit'
end
```

4. 添加下列代码复制到 `SampleHandler.m` 中，将 `APPGROUP` 改为第 1 步创建的 `App Group` 。

```
#import "SampleHandler.h"
#import <TXLiteAVSDK_ReplayKitExt/TXLiteAVSDK_ReplayKitExt.h>
#import <TCICScreenKit/TCICScreenKit.h>
// 注意：此处的 APPGROUP 需要改成上文中的创建的 App Group Identifier。
#define APPGROUP ""

@interface SampleHandler() <TXReplayKitExtDelegate>
@end
@implementation SampleHandler

- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject
*> *)setupInfo {
    [[TXReplayKitExt sharedInstance] setupWithAppGroup:APPGROUP
delegate:self];

    [[TCICScreenKit sharedScreenKit] onScreenKitStarted];
}
- (void)broadcastPaused {
    // User has requested to pause the broadcast. Samples will stop
being delivered.
    [[TCICScreenKit sharedScreenKit] onScreenKitPaused];
```

```
}
- (void)broadcastResumed {
    // User has requested to resume the broadcast. Samples delivery will
resume.
    [[TCICScreenKit sharedScreenKit] onScreenKitResumed];
}
- (void)broadcastFinished {
    [[TXReplayKitExt sharedInstance] finishBroadcast];
    // User has requested to finish the broadcast.
    [[TCICScreenKit sharedScreenKit] onScreenKitFinished];
}
#pragma mark - TXReplayKitExtDelegate
- (void)broadcastFinished:(TXReplayKitExt *)broadcast reason:
(TXReplayKitExtReason)reason
{
    NSString *tip = @"";
    switch (reason) {
        case TXReplayKitExtReasonRequestedByMain:
            tip = @"屏幕共享已结束";
            break;
        case TXReplayKitExtReasonDisconnected:
            tip = @"应用断开";
            break;
        case TXReplayKitExtReasonVersionMismatch:
            tip = @"集成错误（SDK 版本号不相符合）";
            break;
    }
    NSError *error = [NSError
errorWithDomain:NSStringFromClass(self.class) code:0 userInfo:@{
        NSLocalizedFailureReasonErrorKey:tip
    }];
    [self finishBroadcastWithError:error];
}
- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:
 (RPSampleBufferType)sampleBufferType {
    switch (sampleBufferType) {
        case RPSampleBufferTypeVideo:
            [[TXReplayKitExt sharedInstance]
sendVideoSampleBuffer:sampleBuffer];
            break;
        case RPSampleBufferTypeAudioApp:
```

```
            // Handle audio sample buffer for app audio
            break;
        case RPSampleBufferTypeAudioMic:
            // Handle audio sample buffer for mic audio
            break;

        default:
            break;
    }
}
@end
```

5. 对接主 App 端的接收逻辑：目前主 App 中的使用 TCICSDK，已支持系统屏幕分享相关逻辑，只需要业务方配置好 App Group ，在进入课堂前，设置 AppGroup 即可。

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];
roomConfig.userId = "test";
roomConfig.token = "test_token";
roomConfig.classId = 123454;
roomConfig.schoolId = xxxxx;

// 通过KVC方式设置AppGroup
[roomConfig setValue:@"group.com.xx.xxxx" forKey:@"appGroup"];
```

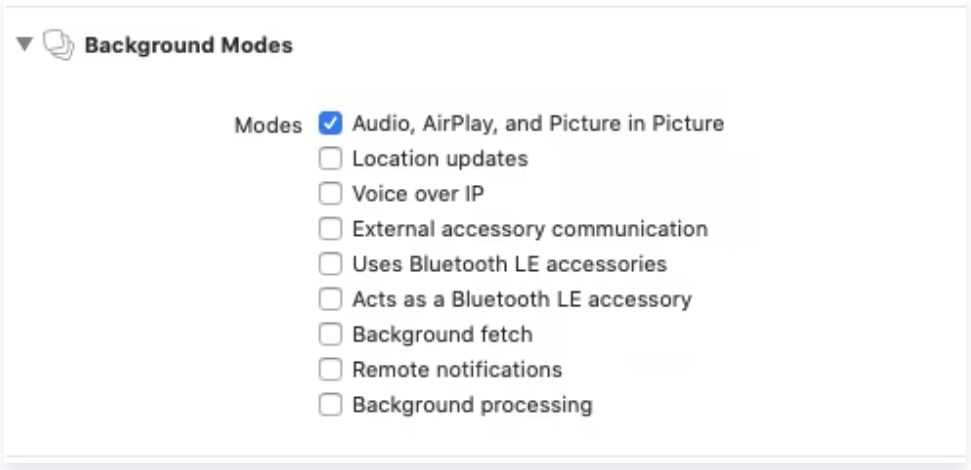
注意事项

1. TCICSDK 中已支持屏幕分享的触发按钮，具体可参见实时音视频文档中 步骤 4：增加屏幕分享的触发按钮（可选），但该功能有限制条件：
   1.1 屏幕分享的触发按钮只支持 iOS12 以上，同时需要创建的工程，不依赖 `Scene` 生命周期，如果代码中已支持 `Scenedelegate`，可参见 Xcode 11 删除 Scenedelegate，进行移除。以 Demo 为例，弹出效

果如下，单击开始直播即可。

1.2 对于 iOS11 的机型，需要业务侧引导用户从远程控制中 长按录屏 进行触发，并选择业务自创建的 `Broadcast Upload Extension` 进行触发，下图以 腾讯会议 为例：

2. 创建的 `Upload Extension` 的 `Deployment target` 配置在 `iOS 11.0`（`Replay Kit` 于 `iOS11` 才开始支持），调试时真机也尽量在 `iOS11` 之后。
3. 主 App 要支持系统级屏幕分享，需要添加 `Background Modes`。



Android 端

> **说明：**
> Android的屏幕分享对版本的要求： Android 5.0 +

除版本要求外，没有其他配置及特殊要求。