实时互动-教育版

客户端集成指引

腾讯云

文档目录

客户端集成指引
Web 和 H5

最近更新时间：2025-10-09 10:09:32

本文主要介绍如何快速将腾讯云实时互动-教育版 LCIC Web/H5 应用快速集成到您的项目中。

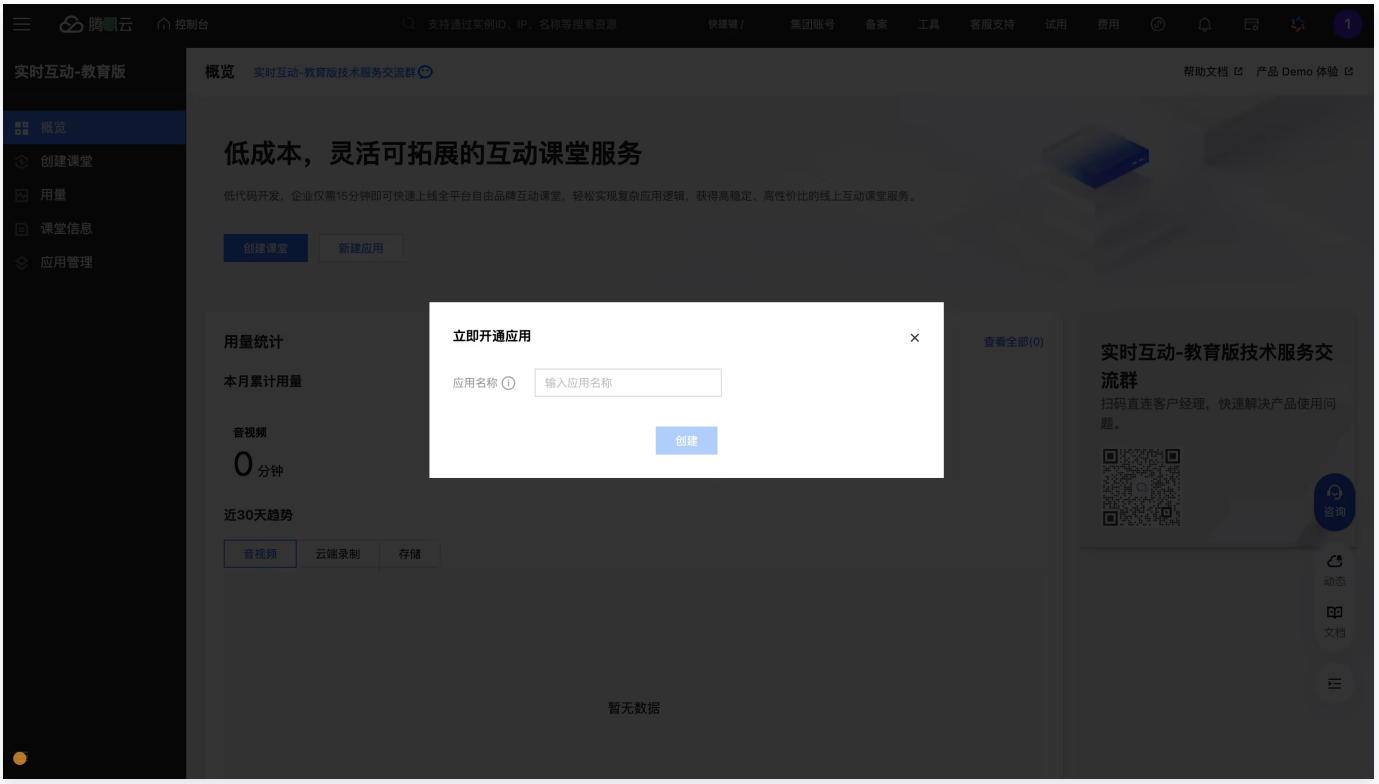
前提条件

1. 您已经 注册腾讯云 账号，并完成 实名认证 。
2. 您使用的桌面或手机浏览器能够支持音视频服务。详细要求可参考 Web 端常见问题 。

操作步骤

步骤一：创建新的应用

1. 登录 实时互动-教育版控制台 ，进入左侧导航栏的**概览**，选择**新建应用**。
2. 若尚未创建应用，则默认进入“创建应用”界面，输入应用名称，例如 TestLCIC。



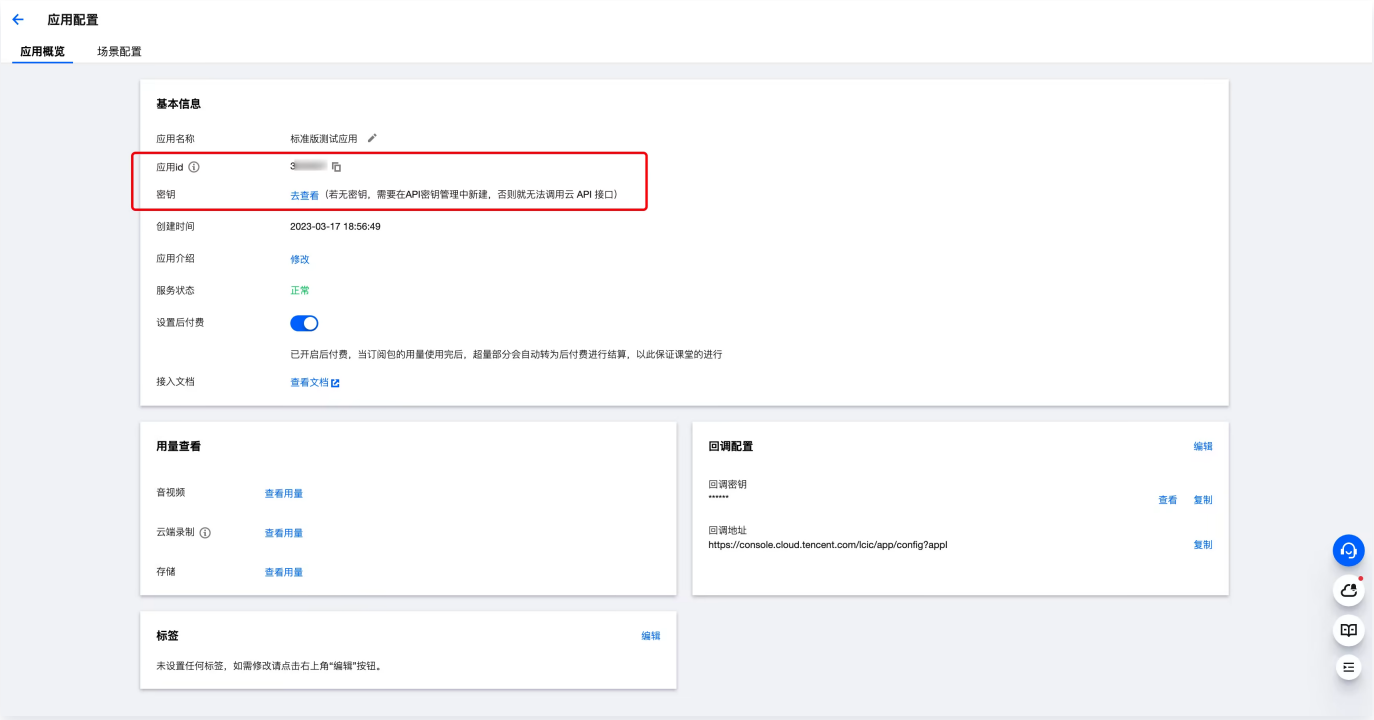
若您已创建应用，可前往**应用管理**中，选择使用已有应用即可。

说明：

- 每个账号可免费领用一个试用版应用，若需创建商用应用，可根据业务需求在 购买页 创建对应版本的应用。
- 应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入 应用管理 > **应用配置**，获取 SDKAppId。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中新建，具体可参见 访问密钥管理 。



步骤三：获取进入课堂所需参数

1. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(`userid`)信息。
2. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 `token` 信息。
3. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(`classid`)信息。
4. 明确需要集成的课堂版本：latest。

5. 其中 `scene` 、 `debugjs` 、 `debugcss` 为非必填参数，在需要自定义 UI 时才需设置，具体可参见 自定义 UI 集成 。其中 `debugjs` 和 `debugcss` 只用于自定义布局、组件时的调试，且只支持通过 `localhost` 或 `127.0.0.1` 的地址访问，在发布阶段请勿使用此参数。
6. `lng` 、 `location` 、 `layout` 也是非必填参数，业务侧可自行判断是否需要传入，不传则使用默认值，其中 `layout` 参数只有在教室布局为视频+文档布局(`videodoc`)时才生效。

字段	类型	含义	备注	必填
userid	string	用户名	通过 RegisterUser 接口获取。	是
classid	string	课堂 ID	通过 CreateRoom 接口创建返回获取。	是
token	string	后台鉴权参数	通过 LoginUser 接口获取。	是
version	string	课堂版本号	通过发布日志选择对应版本。说明：互动课堂客户端版本，建议使用 latest。	是
scene	string	场景名称	用于区分不同的定制布局，通过 SetAppCustomContent 接口配置，默认为 `default` 。	否
debugjs	string	自定义 UI 的 JS 链接	通过自定义 UI 集成方式获取。	否
debugcss	string	自定义 UI 的 CSS 链接	通过自定义 UI 集成方式获取。	否
role	string	进入课堂角色，默认空	可选参数 supervisor(巡课/内容审查)，只有已注册应用内巡课用户才有权限。	否
lng	string	语言参数，默认 zh-CN	当前支持中文(简体)、中文(繁体)、English、韩语、日语、阿拉伯语、越南语、印尼语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi、id。	否
micAutoOpen	number	麦克风默认开关状态	针对上台的用户，此参数有效，可控制进入课堂的麦克风默认开关。 • 0：关闭。 • 1：开启。	否
cameraAuto	numb	摄像头默认开关状态	针对上台的用户，此参数有效，可控制进入课堂的摄像头默认开关。	否

Open	er		• 0：关闭。 • 1：开启。	
location	boolean	是否上报经纬度位置信息	默认 false 不上报。	否
layout	string	页面布局	默认顶部布局(top)，当前仅视频文档模式有效，支持双排布局（double）、右侧布局(right)、左侧布局(left)、三分布局(three)。	否
boardColor	string	白板颜色	白板颜色设置，默认为：#182E25 ，支持 Hex 格式，也支持 rgba(0, 0, 0, .3)设置。	否
noEndClass	boolean	禁用下课	助教进房时带上这个参数，在助教点击退出时，将隐藏「下课」，仅展示「离开」按钮。	否
back_url	string	退出课堂回调地址	默认为空，单击返回或退出课堂时需要回调的页面地址。参数需要使用 encodeURIComponent 编码。	否

步骤四：进入课堂

将刚才获取到的参数，按以下规范拼接出用户进入课堂的 URL，通过跳转此 URL 链接即可实现 Web 端的集成。

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

具体链接如下所示：

```
https://class.qcloudclass.com/latest/class.html?
userid=12345&token=yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwN
zQwMjEsImlhdCI6MTY3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JY
WpleVh2RWwiLCJzY2hvb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCN
VhCczZHT1BnVXpweUgifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII&classi
d=368507569&schoolid=23456
```

说明：

- 在拼接的 URL 中 `userid` 与 创建课堂 指定的老师 ID（ `teacherid` ）是一致的，则当前用户为老师。若与当前课堂的助教 ID（ `assistantid` ）一致，则为助教，否则为学生。
- URL 链接中的 schoolid 在 控制台应用管理-应用 id 处获取。

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://class.qcloudclass.com/latest/class.html?
userid=12345&token=xxx"
>
```

高级功能

自定义 UI 集成

实时互动-教育版 LCIC Web/H5 目前还提供了自定义 UI 的集成方案。用户可自定义业务侧课中的布局及样式，通过 自定义 UI 部分 可以获取到业务侧的 JS 及 CSS 链接，将 `debugjs` 及 `debugcss` 参数拼接到上方的链接上即可（此参数只用于本地 localhost 调试），如下所示：

```
https://class.qcloudclass.com/latest/class.html?
userid=12345&token=yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwN
zQwMjEsImlhdCI6MTY3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JY
WpleVh2RWwiLCJzY2hvb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCN
VhCczZHT1BnVXpweUgifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII&classi
d=368507569&schoolid=23456&debugjs=http://localhost:443/demo/dist/myLib.
umd.min.js&debugcss=http://localhost:443/demo/dist/myLib.css
```

当自定义 JS 与 CSS 调试完成后，可通过云 API 接口 SetAppCustomContent 或 控制台应用管理 > 应用配置 > 场景配置将场景与自定义的 JS、CSS 链接（不可使用带端口的地址， 否则会被拦截）进行绑定，在进入课堂时将 `scene` 参数拼接到 URL 上，即可加载对应场景的布局及组件。在涉及多种班型、多种布局时，业务侧可根据此参数实现场景的切换。如下所示：

```
http://class.qcloudclass.com/latest/class.html?
userid=12345&token=yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwN
zQwMjEsImlhdCI6MTY3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JY
WpleVh2RWwiLCJzY2hvb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCN
VhCczZHT1BnVXpweUgifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII&classi
d=368507569&schoolid=23456&scene=default
```

自定义业务域名

在课中页面时，若业务侧想隐藏课堂域名，只显示业务域名，可通过 内容分发网络(CDN)控制台 新建业务域名，并回源到课堂域名即可，详细流程请参考 自定义业务域名 。

> **注意：**
> 自定义域名的情况下，不支持使用虚拟背景和美颜相关的能力。如业务中涉及，可咨询 在线客服 寻求帮助。

其他相关文档

- LCIC API
- 自定义 UI 集成
- Web 常见集成问题

小程序插件

最近更新时间：2025-12-08 11:20:31

本文主要介绍如何将实时互动-教育版 LCIC 小程序插件集成到您的小程序项目中。可单击查看 小程序插件功能介绍 。

插件说明

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

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

注意：

- 微信小程序的主体必须为非个人主体类型，否则无法使用直播功能。以上表格**仅提供参考**，详细的微信小程序类目及申请资质要求需以微信最新的 微信非个人主体小程序开放的服务类目 为准。
- 微信小程序的类目即为微信小程序的服务场景，在小程序后台的**设置 > 基本设置 > 服务类目**中，可以选择符合小程序功能的类目。所选类目需符合小程序的实际应用场景，否则在提交审核后会被驳回申请。
- 您申请小程序标签的公司主体**无需**与您接入实时互动-教育版 LCIC 的腾讯云账号的主体一致。

环境要求

- 微信 App iOS 最低版本要求：7.0.9
- 微信 App Android 最低版本要求：7.0.8

- 小程序基础库最低版本要求：2.10.0

调用流程图

操作步骤

步骤一：自行开发业务小程序页面

业务侧需自行实现课前部分，包括创建课堂、进入课堂、直播页部分组件等部分，具体如 小程序 Demo 所示。
自行开发上述小程序页面前需要了解：

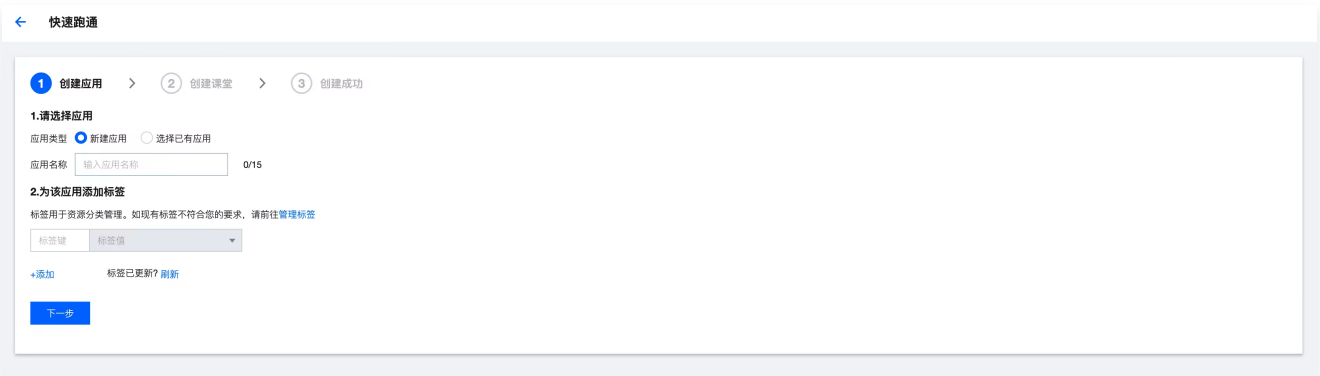
1. 创建新的应用。
2. 获取 SDKAppId 和密钥（SecretKey）。
3. 获取进入课堂所需参数，详情参见 Web 。

1. 创建新的应用

1.1. 登录 实时互动-教育版 控制台，在左侧导航栏选择**快速跑通应用**。
1.2. 默认进入“创建应用”界面，应用类型可选择“**创建新应用**”，输入应用名称，例如 TestLCIC。
若您已创建应用，应用类型项可单击“选择已有应用”。

> **说明：**
> 每个账号可免费领用一个试用版应用，若需创建商用版应用，可根据业务需求在 购买页 选择并购买对应版本。

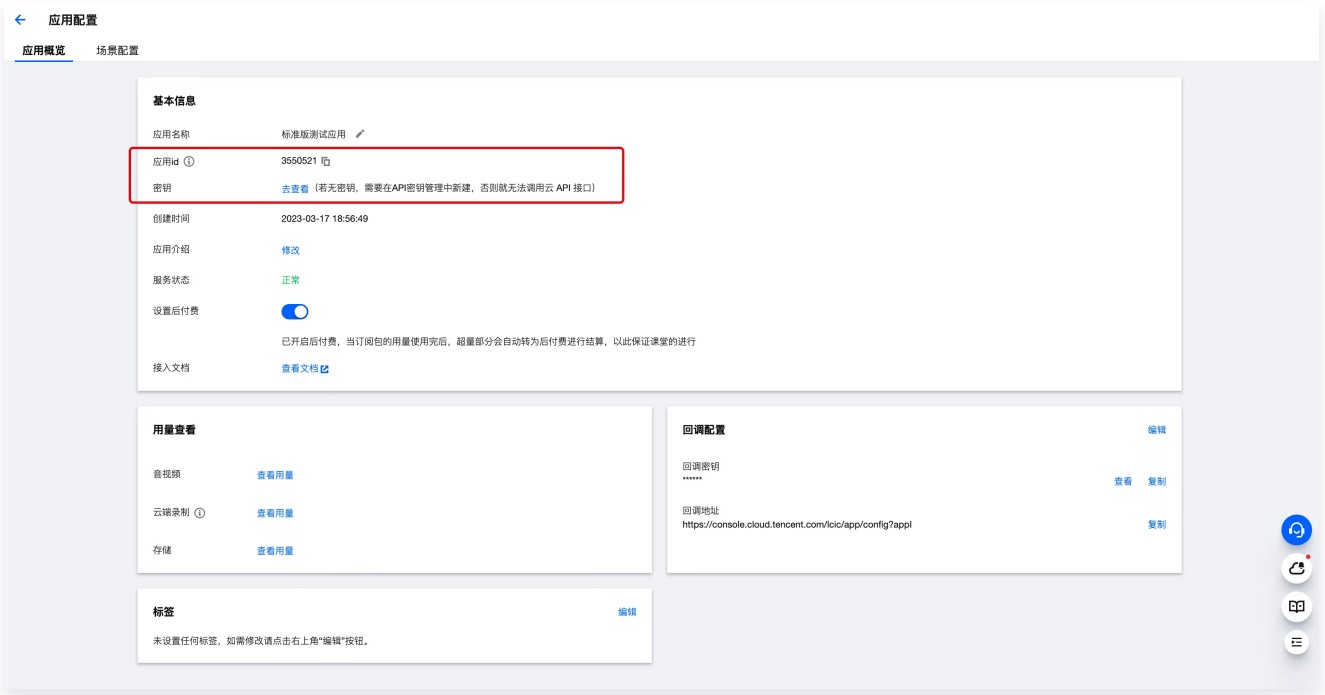
1.3. 添加或编辑标签（可选），单击**下一步**。



> **说明：**
> - 应用名称只允许使用下划线、数字或中英文字符。
> - 标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 LCIC 应用，这时，企业可以通过给 LCIC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

2. 获取 SDKAppId 和密钥(SecretKey)

2.1. 进入**应用管理 > 应用配置**，获取 SDKAppId 。
2.2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中进行新建，具体可参考 访问密钥管理 。



3. 获取进入课堂所需参数

3.1. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(`userid`)信息。
3.2. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 `token` 信息。
3.3. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(`classid`)信息。
3.4. 选择需要集成的课堂版本，一般业务侧集成最新版本即可。
3.5. 其中 `scene` 、 `debugjs` 、 `debugcss` 为非必填参数，在需要自定义 UI时才需设置，具体可参考自定义 UI 集成。其中 `debugjs` 和 `debugcss` 只用于自定义布局、组件时的调试，且只支持通过 `localhost` 或 `127.0.0.1` 的地址进行访问，在发布阶段请勿使用此参数。
3.6. `lng` 、 `location` 、 `layout` 也是非必填参数，业务侧可自行判断是否需要传入，不传则使用默认值，其中 `layout` 参数只有在教室布局为视频+文档布局(`videodoc`)时才生效。

字段	类型	含义	备注	必填

userid	string	用户名	通过 RegisterUser 接口获取。	是
classid	string	课堂 ID	通过 CreateRoom 接口创建返回获取。	是
token	string	后台鉴权参数	通过 LoginUser 接口获取。	是
version	string	课堂版本号	通过发布日志选择对应版本。	是
scene	string	场景名称	用于区分不同的定制布局，通过 SetAppCustomContent 接口配置。	否
debugjs	string	自定义 UI 的 JS 链接	通过自定义 UI 集成方式获取。	否
debugcss	string	自定义 UI 的 CSS 链接	通过自定义 UI 集成方式获取。	否
lng	string	语言参数，默认 zh-CN	当前支持中文(简体)、中文(繁体) 、English、韩语、日语、阿拉伯语、越南语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi。 还支持西班牙语、法语、德语、泰语、马来语、印尼语等多语种，可联系您的腾讯云商务经理或产品经理来了解。	否
location	boolean	是否上报经纬度位置信息	默认 false 不上报。	否
layout	string	页面布局	默认顶部布局(top)，当前支持双排布局（double）、右侧布局(right)、左侧布局(left)、三分布局(three)。	否
back_url	string	回调地址	支持退出课堂回跳到业务侧地址如课后调查，用户列表等 页面版本1.7.3+。	否

步骤二：课程页面通过小程序插件接入

若是首次使用小程序插件，请阅读了解微信官方文档小程序 使用插件 内容。

1. 添加插件

点击 添加插件，等待插件管理员通过后，方可在小程序中使用相应的插件。如有需要可扫码进群与管理员沟通：

2. 引入插件

需要在业务小程序内，根据以下步骤进行配置。

2.1. 在 app.json 的 plugins 字段的对象里添加插件配置：

```
    "plugins": {
        ...
        "tcic-plugin": {
          "version": "1.8.0", // 使用的插件版本
          "provider": "wxbc2aedd3838d78cd", // 插件小程序appid
          "export": "tcicPluginConfig.js", // 插件可接收的参数配置信息
          "genericsImplementation": {
            "interactive": { // 支持您的小程序中组件覆盖插件的模块，从而
实现自定义
              "interactive-class-
info": "components/live/interactive-class-info"
            }
          }
        }
        ...
      },
```

2.2. 插件配置里需要一个 tcicPluginConfig.js 的文件，和 app.json 同级（或者放置其他目录也可，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, // 用于传递IM消息
  loginUrl: '/pages/index/login', // 小程序登录页地址 [业务侧自行实现]
  homeUrl: '/pages/index/home', // 小程序首页地址 [业务侧自行实现]
  Config: { // 页面内可配置项
    LIVE: {
      MAX_TEXT_NUM: 200, // IM文本最大长度
      QUICK_INPUT_TEXTS: [],
      RECORD_TIPS: '您可以到课程列表中查看回放', // 课程结束有回放时的提示内容
    },
    IMG: {
      MAX_SIZE: 20 * 1024 * 1024, // 图片大小 20MB
    },
    ENABLE_TROPHY: true, // 是否启用奖励特效 [1.7.19新增特效]
    HIDE_USER_LIST_BAR: false, // 隐藏用户列表栏 [1.7.12支持]
    ORIENTATION: 'portrait', // 横屏或竖屏打开，其他可选值landscape [1.7.8支持]
    NAV_TITLE: '实时互动-教育版', // 导航栏标头，可为空字符串
    VIDEO_WATER_MARK: '', // 视频左上角水印，可为空字符串
  },
};
```

```
// app.js

const events = new Map();
App({
  onLaunch() {},
  onShow() {},
  ...
  // setValue 需要从小程序app 里面传递给插件
  setValue(key, data) {
    const value = events.get(key);
    if (Array.isArray(value)) {
      value.forEach((e) => {
        const callback = e.callback;
        callback(data);
      });
```

```
    }
  });
```

2.3. 直播页部分组件需要由小程序提供，如 class-info、landscape-im-container 等。参考代码可以从 代码片段 获取或咨询管理员。

```
// 参考代码片段时，接入方需要额外注意的地方
// 1、app.js中上挂载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、配置request白名单
// https://tcic-api.qcloudclass.com;
// https://tcic-demo.qcloudclass.com;
// * 2022年12月之后接入业务不需要以下配置
// https://tcic-api.qcloudtiw.com;
// https://tcic-demo.qcloudtiw.com;
// https://tcic-demo-login.qcloudclass.com;

// 5、直播简介
// components/live/class-info.js 修改 getClassInfo
// 6、分享内容配置
// components/live/class-info.js 修改 tcicPlugin.setShareInfo
// 7、直播画面水印配置
// components/live/class-info.js 修
改 tcicPlugin.setValue('watermark', {...})
```

2.4. 跳转到插件页面。

```

```

```
// 按钮形式
<navigator url="plugin://tcic-plugin/interactive">
    跳转到课中页面
</navigator>

// API跳转
 goToLiveRoom () {
  // 组装必传参数
  const params = {
    classid: 222,
    userid: 'xxx',
    token: 'yyy',
    type: 'interactive' // type值为 interactive 或者 live
  };
  // 后续代码无需更改
  const arr = [];
  for (const [key, val] of Object.entries(params)) {
    arr.push(`${key}=${val}`);
  }
  const url = `plugin://tcic-plugin/${params.type}?
${arr.join('&')}`;
  wx.navigateTo({
    url,
  });
```

参数说明：

参数	类型	描述	备注
classid	String	课堂 id（即 RoomId）	通过 CreateRoom 接口创建返回获得（RoomId）
userid	String	用户 id	通过 RegisterUser 接口获得（UserId）
token	String	token	通过 LoginUser 接口获得（Token）
type	String	课堂类型 (interactive | live)	live 为公开直播课，不能互动，interactive 为互动课堂

常见问题

插件体积较大，容易导致主包发布超出 2M 限制怎么解决?

接入方可以考虑在分包里使用插件，需要从分包的页面跳转进插件页。

- 关于“分包中引入插件，会导致分包中页面样式不生效”的问题，可参见 微信开放社区反馈 。

  由于开发工具的问题，目前最优解决方案是：将分包样式写在 app.wxss 文件中，作为全局样式。
- 分包之后报【 tcicPluginConfig.js is not defined 】之类的错误。

  请先确认导入路径、名称无误，如若不行，则将 tcicPluginConfig.js 文件复制一份到分包文件根目录下。

Android 原生内核

最近更新时间：2025-12-05 18:04:01

开发环境要求

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

前置准备

您已 注册腾讯云 账号，并完成 实名认证 。

步骤一：创建新的应用

1. 登录 实时互动-教育版 控制台，进入左侧导航栏的**概览**，选择**新建应用**。
2. 若尚未创建应用，则默认进入“创建应用”界面，输入应用名称，例如 TestLCIC。

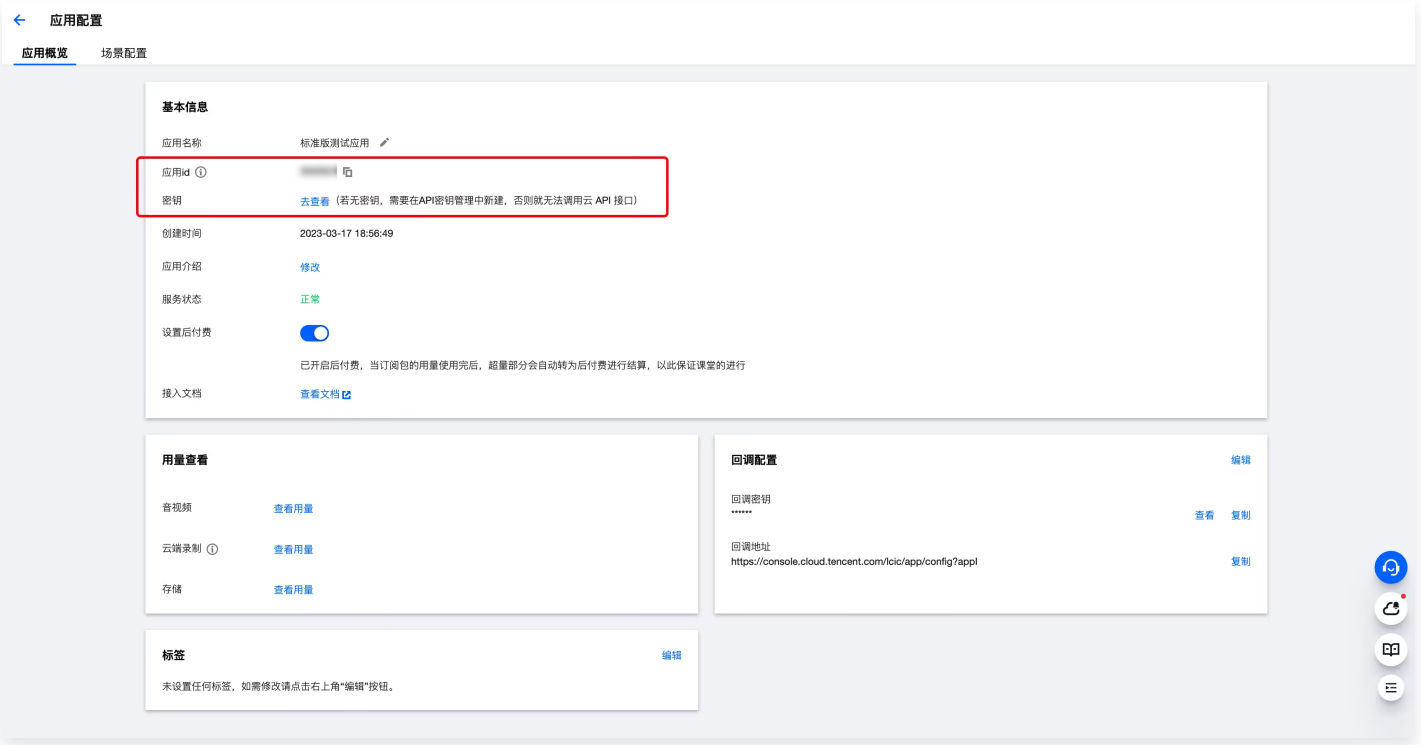
若您已创建应用，可前往**应用管理**中，选择使用已有应用即可。

说明：

- 移动端需要购买旗舰版或企业尊享版后方可接入。若需创建商用应用，可根据业务需求在 购买页 创建对应版本的应用。
- 应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入 应用管理 > **应用配置**，获取应用 ID（SDKAppId）。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中进行新建，具体可参见 访问密钥管理 。



步骤三：获取进入课堂所需参数

1. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(userId)信息。
2. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 token 信息。
3. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(classId)信息。

字段	类型	必填	含义	备注
userid	string	是	用户名。	通过 RegisterUser 接口获取。
classid	string	是	课堂 ID。	通过 CreateRoom 接口创建返回获取。
token	string	是	后台鉴权参数。	通过 LoginUser 接口获取。

role	string	否	进入课堂角色，默认空。	可选参数 supervisor(巡课/内容审查)，只有已注册应用内巡课用户才有权限。
lng	string	否	语言参数，默认 zh-CN。	当前支持中文(简体)、中文(繁体)、English、韩语、日语、阿拉伯语、越南语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi。
location	boolean	否	是否上报经纬度位置信息。	默认 false 不上报。
layout	string	否	页面布局。	默认顶部布局(top)，当前仅视频文档模式有效，支持双排布局（double）、右侧布局(right）、左侧布局(left)、三分布局(three)。
boardColor	string	否	白板颜色。	白板颜色设置，默认为：#182E25，支持 Hex 格式，也支持 rgba(0, 0, 0, .3)设置。
noEndClass	boolean	否	禁用下课。	助教进房时带上这个参数，在助教点击退出时，将隐藏「下课」，仅展示「离开」按钮。

快速跑通

根据下面的步骤您可以快速跑通项目，具体代码也可以参见 Demo 。

步骤一：导入 SDK

1. 在您的 `Android` 项目中的 `settings.gradle` 中添加：

```
maven("https://storage.googleapis.com/download.flutter.io")
maven("https://android.qcloudclass.com/repo")
```

2. 在您的 `Android` 项目中的 `build.gradle` 文件中添加依赖，并且设置 `compileSdk` 。

```
compileSdk = 35

implementation("com.qcloudclass.tcic_client_module:flutter_release:1.0
.6")
implementation("com.qcloudclass:tcic:1.0.14")
```

```
implementation("io.flutter:flutter_embedding_release:1.0.0-
18b71d647a292a980abb405ac7d16fe1f0b20434")
```

3. 在您的 `AndroidManifest.xml` 中的 `application` 添加 `tools:replace="android:allowBackup"`。

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    tools:replace="android:allowBackup" // 添加这一行
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Tcic_android_simple_demo"
    tools:targetApi="31">
   ... 其他代码
```

步骤二：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"/>
```

步骤三：初始化 SDK

在您的页面首次进入的时候需要初始化 TCIC SDK，您可以根据您的实际业务来处理。

```
TCICManager.initialize(context);
```

步骤四：设置回调事件

如果您需要监听课堂相关的事件，可添加相关回调，同初始化一样，您只需要添加一次即可。

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

    override fun afterExitedClass() {
        Log.d(TAG, "afterExitedClass called from Flutter")
        runOnUiThread {
            Toast.makeText(ctx, "已退出课堂，关闭页面",
Toast.LENGTH_SHORT).show()
            TCICManager.closeTCICActivity();
        }
    }

    override fun onJoinedClassFailed() {
        Log.d(TAG, "onJoinedClassFailed called from Flutter")
        runOnUiThread {
            Toast.makeText(ctx, "加入课堂失败", Toast.LENGTH_SHORT).show()
            TCICManager.closeTCICActivity();
        }
    }

    override fun onKickedOffClass() {
        Log.d(TAG, "onKickedOffClass called from Flutter")
        runOnUiThread {
            Toast.makeText(ctx, "被踢出课堂", Toast.LENGTH_SHORT).show()
        }
    }
```

```
    override fun onMemberJoinedClass(data: Map<*, *>) {
        Log.d(TAG, "onMemberJoinedClass: $data")
        runOnUiThread {
            Toast.makeText(
                ctx,
                "成员加入: $data",
                Toast.LENGTH_SHORT
            ).show()
        }
    }

    override fun onMemberLeaveClass(data: Map<*, *>) {
        Log.d(TAG, "onMemberLeaveClass: $data")
        runOnUiThread {
            Toast.makeText(
                ctx,
                "成员离开: $data",
                Toast.LENGTH_SHORT
            ).show()
        }
    }

    override fun onRecivedMessage(message: Map<*, *>) {
        Log.d(TAG, "onRecivedMessage: $message")
        runOnUiThread {
            Toast.makeText(
                ctx,
                "收到消息: $message",
                Toast.LENGTH_SHORT
            ).show()
        }
    }

    override fun onError(errorCode: String, errorMsg: String) {
        Log.e(TAG, "onError: $errorCode - $errorMsg")
        runOnUiThread {
            Toast.makeText(
                ctx,
                "错误: $errorMsg",
                Toast.LENGTH_SHORT
```

```
            ).show()
        }
    }
})
```

步骤五：进入课堂页面

在前置准备中，您已了解如何获取 token、userId 等进入课堂的必要参数，您只需要根据您的实际业务，填入这些参数即可。

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) { val context
= LocalContext.current

    // 这里用 Column 来垂直排列文本和按钮
    androidx.compose.foundation.layout.Column(modifier = modifier) {

        Button(onClick = {
            val token = "";
            val classId = ""
            val userId = ""
            val role = 1 /// 用户角色，0: 学生,1: 老师, 3: 助教, 4: 巡课
            val config = TCICConfig(token, classId, userId);
            TCICManager.setConfig(config);
            // 打开activity
            context.startActivity(TCICManager.getTCICIntent(context));
        }) {
            Text(text = "点击我进房")
        }
    }
}
```

步骤六：页面自定义 view

您同时可以自定义课堂中的某些部分，例如 Header 组件， Message 组件的消息体等；例如自定义 Header 组件 Left 部分。下文以构造 `Header` 组件部分为例。

创建 viewCreator

```
public class HeaderLeftViewCreator implements NativeViewCreator {
```

```
    @Override
    public View createView(Context context, int id, Object args) {
        // 水平布局：文本 + info 图标
        LinearLayout layout = new LinearLayout(context);
        layout.setOrientation(LinearLayout.HORIZONTAL);
        layout.setGravity(Gravity.CENTER_VERTICAL);
        layout.setBackgroundColor(Color.TRANSPARENT);

        // 创建文本
        TextView textView = new TextView(context);
        String text = "Header Left 原生View";
        textView.setText(text);
        textView.setTextSize(12); // 12px
        textView.setTextColor(Color.WHITE);
        textView.setGravity(Gravity.CENTER_VERTICAL);

        LinearLayout.LayoutParams textParams = new
LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT);
        textView.setLayoutParams(textParams);
        layout.addView(textView);

        // 创建 info 图标
        ImageView infoIcon = new ImageView(context);

infoIcon.setImageResource(android.R.drawable.ic_menu_info_details); // 使
用系统 info 图标
        infoIcon.setColorFilter(Color.WHITE); // 设置图标颜色为白色

        // 在 createView 里
        LinearLayout.LayoutParams iconParams = new
LinearLayout.LayoutParams(
                dp2px(context, 32), dp2px(context, 32)); // 32dp x 32dp
        iconParams.setMargins(8, 0, 0, 0); // 左边距 8px
        infoIcon.setLayoutParams(iconParams);
        layout.addView(infoIcon);

        return layout;
```

```
    }
    // dp 转 px 工具
    private int dp2px(Context context, float dp) {
        return (int) (dp *
context.getResources().getDisplayMetrics().density + 0.5f);
    }

    @Override
    public void disposeView(View view) {
        // 可以在这里处理 View 的清理工作
        // 例如取消监听器、释放资源等
    }
}
```

将 View Creator 注册到 SDK

所有组件的配置以及自定义配置都只需要在 `TCICConfig` 中进行配置即可。

```
val headerComponentConfig = TCICHeaderComponentConfig();
headerComponentConfig.setHeaderLeftBuilder(::HeaderLeftViewCreator)
val token = "";
val classId = ""
val userId = ""
val role = 1 /// 用户角色，0: 学生,1: 老师, 3: 助教, 4: 巡课
val config = TCICConfig(token, classId, userId, role);
config.headerComponentConfig = headerComponentConfig;
TCICManager.setConfig(config);
context.startActivity(TCICManager.getTCICIntent(context));
```

这样您就获得了一个自定义 `Header` 组件的页面。

其他

TCICCallback 接口

```
public interface TCICCallback {
    /**
     * 加入课堂成功
     */
    void onJoinedClassSuccess();
```

```
    /**
     * 退出课堂后
     */
    void afterExitedClass();

    /**
     * 加入课堂失败
     */
    void onJoinedClassFailed();

    /**
     * 被踢出课堂
     */
    void onKickedOffClass();

    /**
     * 成员加入课堂
     * @param userId 用户信息
     */
    void onMemberJoinedClass(Map userId);

    /**
     * 成员离开课堂
     * @param userId 用户信息
     */
    void onMemberLeaveClass(Map userId);

    /**
     * 收到消息
     * @param messageJson 消息内容
     */
    void onRecivedMessage(Map messageJson);

    /**
     * 发生错误
     * @param errorCode 错误代码
     * @param errorMsg 错误信息
     */
    void onError(String errorCode, String errorMsg);
```

```
}
```

头部组件配置 (TCICHeaderComponentConfig)

```
TCICHeaderComponentConfig headerConfig = new
TCICHeaderComponentConfig();

// 设置左侧视图
headerConfig.setHeaderLeftBuilder(new HeaderLeftViewCreator());

// 设置头部视图
headerConfig.setHeaderBuilder(new HeaderViewCreator());

// 设置头部操作视图
headerConfig.setHeaderActionsBuilder(new HeaderActionCreator());

// 设置右侧视图
headerConfig.setHeaderRightBuilder(new HeaderRightViewCreator());
```

消息组件配置

```
TCICMessageComponentConfig messageConfig = new
TCICMessageComponentConfig();

// 设置消息头部视图
messageConfig.setMessageHeaderBuilder(new MessageHeaderViewCreator());

// 设置消息气泡视图
messageConfig.setMessageBubbleBuilder(new MessageBubbleViewCreator());

// 设置消息项视图
messageConfig.setMessageItemBuilder(new MessageItemViewCreator());

// 设置消息行视图
messageConfig.setMessageRowBuilder(new MessageRowViewCreator());
```

音视频组件配置

```
TCICVideoComponentConfig videoConfig = new TCICVideoComponentConfig();

// 设置视频操作视图
videoConfig.setVideoActionBuilder(new VideoActionViewCreator());

// 设置视频浮动视图
videoConfig.setVideoFloatBuilder(new VideoFloatViewCreator());
```

自定义视图

```
public class CustomViewCreator implements NativeViewCreator {
    @Override
    public View createView(Context context) {
        // 创建自定义视图
        View customView =
LayoutInflater.from(context).inflate(R.layout.custom_view, null);
        return customView;
    }
}
```

注册自定义视图

```
TCICManager.registerNativeViewCreator("custom_view_type", new
CustomViewCreator());

// 检查视图是否已注册
boolean isRegistered =
TCICManager.isViewCreatorRegistered("custom_view_type");

// 获取所有已注册的视图类型
List<String> registeredTypes = TCICManager.getRegisteredViewCreators();
```

技术支持

如果您在使用过程中遇到问题，可以通过以下方式获取技术支持：

- 腾讯云实时互动-教育版官方文档
- 腾讯云开发者社区
- GitHub Issues

Web 内核

最近更新时间：2025-12-09 14:43:52

开发环境要求

- Android Studio 3.0+
- Android 6.0（23）及以上系统

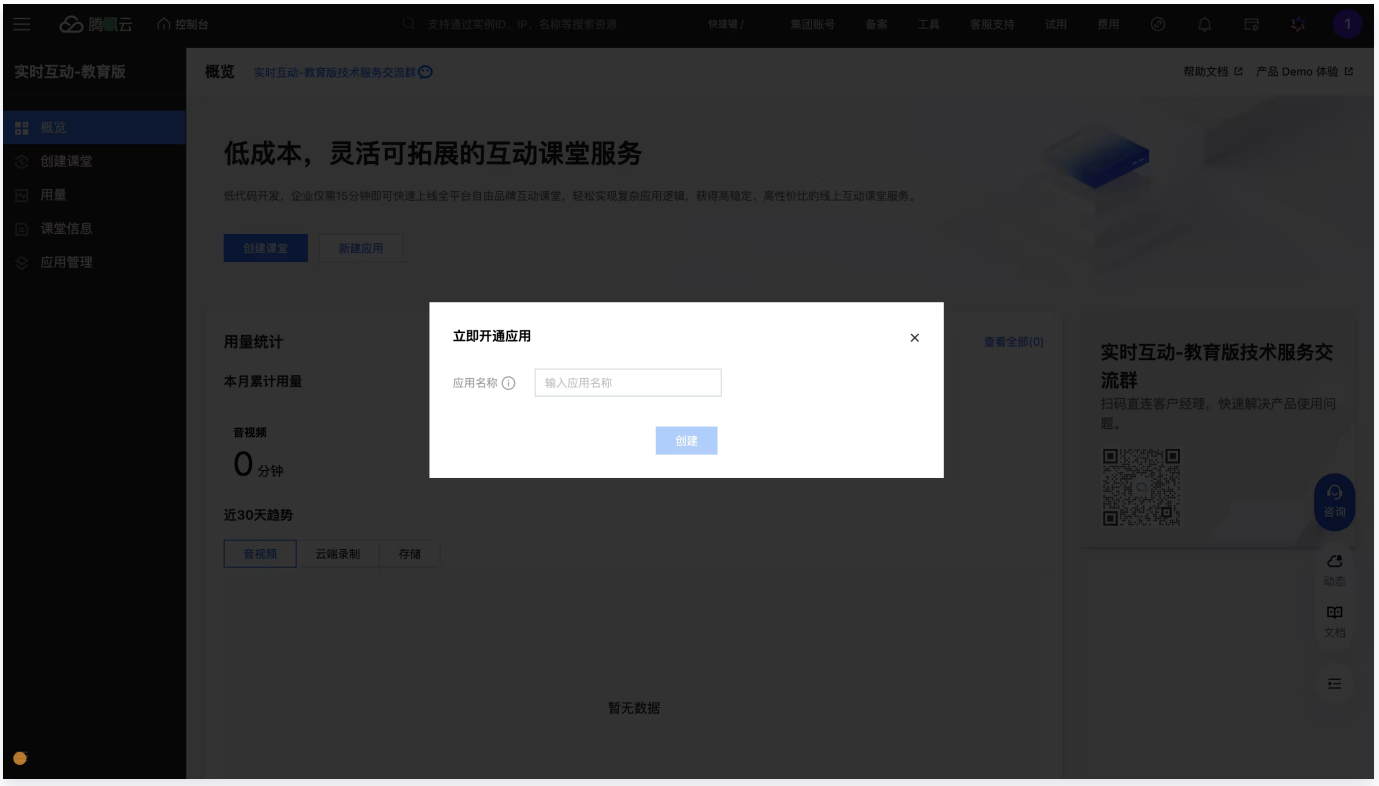
前提条件

您已 注册腾讯云 账号，并完成 实名认证 。

操作步骤

步骤一：创建新的应用

1. 登录 实时互动-教育版 控制台，进入左侧导航栏的**概览**，选择**新建应用**。
2. 若尚未创建应用，则默认进入“创建应用”界面，输入应用名称，例如 TestLCIC。

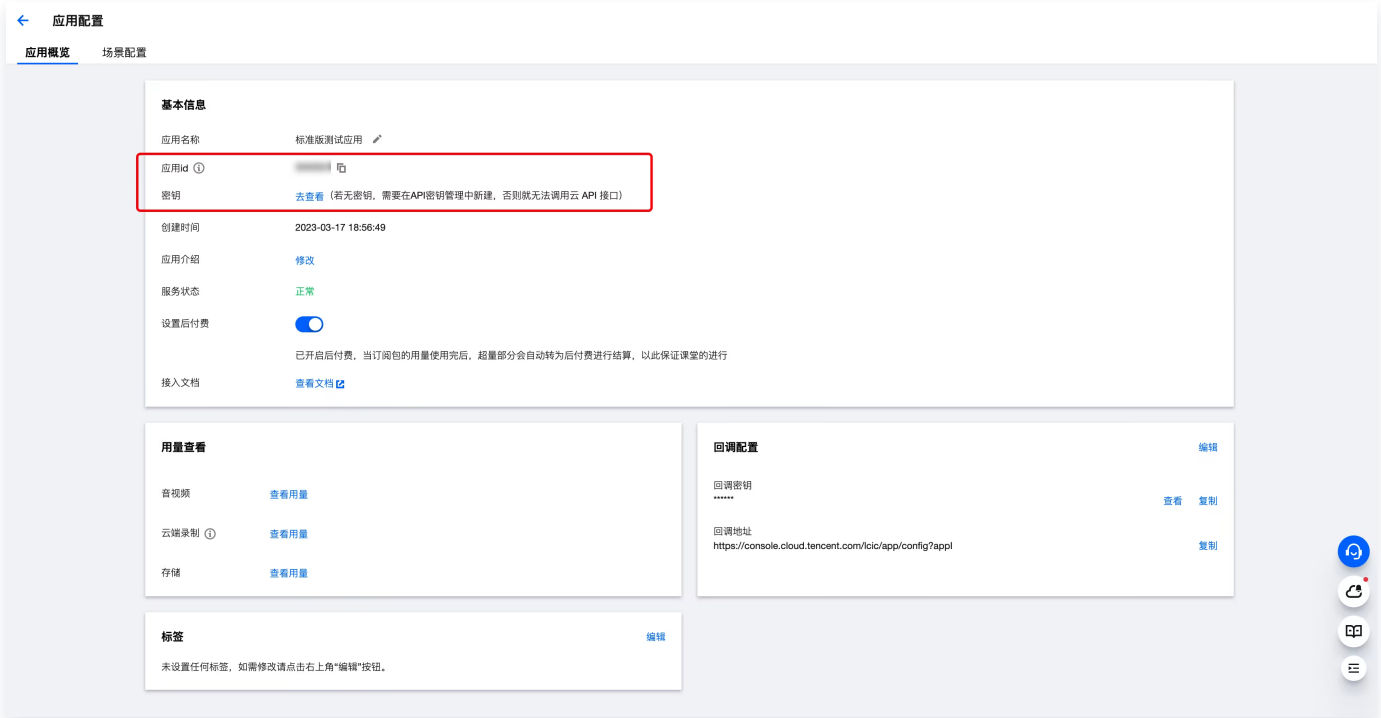


若您已创建应用，可前往**应用管理**中，选择使用已有应用即可。

> **说明：**
> - 移动端需要购买旗舰版或企业尊享版后方可接入。若需创建商用应用，可根据业务需求在 购买页 创建对应版本的应用。
> - 应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入**应用管理 > 应用配置**，获取 SDKAppId 。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中进行新建，具体可参见 访问密钥管理 。



步骤三：导入 SDK

Gradle 工程 SDK 远程构建

LCIC SDK 已经发布到 Maven Central 库，您可以通过配置 Gradle 自动下载更新。

1. 应用模块的 build.gradle 中，在 dependencies 中添加 LCIC SDK 的依赖。

```
dependencies {
    // LCICSDK 组件
    implementation 'com.tencent.edu:TCICSDK:1.8.21'
}
```

2. 在 defaultConfig 中，指定 App 使用的 CPU 架构。

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

> **说明：**
> 目前 LCIC SDK 支持 armeabi-v7a 和 arm64-v8a，能够根据业务需求进行灵活配置。支持不同架构时，包体大小可能会有所不同。

3. 在 compileOptions 中，使用 JDK 1.8 编译。

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

4. 如果 AGP 大于4.2.0，需要在应用模块的 build.gradle 中，配置以下内容：

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

5. 单击 Sync Now，自动下载 SDK 并集成到工程里。

步骤四：配置清单文件

在 AndroidManifest.xml 中配置 App 的权限，LCIC SDK 需要以下权限：

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

> ⚠ **注意：**
> 当 minSdkVersion ≥ 23时，在 AndroidManifest.xml 中 application 添加 `android:extractNativeLibs="true"` 参数。

步骤五：设置混淆规则（可选）

如设置了混淆，需在 proguard-rules.pro 文件，将 LCIC SDK 相关类加入不混淆名单：

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

步骤六：SDK 授权申请

需要您提交 腾讯云工单，向我们发送 SDK 权限申请。请按以下模板提供对应信息。在信息确认无误的情况下，我们将会在1个工作日完成。

> ⚠ **注意：**
> - 一个旗舰版仅支持授权一个正式包名，请确认无误后发送相关信息。
> - 包名用于 X5 内核以及快直播播放器签名授权，请提供所需授权的正式应用的 App Name、Package Name 和 Bundle ID 信息。

分类	说明
问题标题	实时互动-教育版 Android SDK 授权申请
问题主要内容	公司名称。如，xxx 有限公司。
	个人姓名
	联系方式
	App Name
	Package Name（Android）

	Bundle ID（iOS）

步骤七：初始化 X5 内核

X5 内核相对于系统 WebView，具有兼容性更好，速度更快等优势。Android 实时互动-教育版 SDK 的组件实现依赖于 X5 内核的 WebView。现提供 X5 内核静态集成方式，能提升 X5 内核加载成功率且无需进程重启即可生效。

1. 检查同意隐私政策协议。

> ⚠ **注意：**
> 建议在同意隐私政策协议之后，再调用初始化 X5 内核的方法，以免上架应用时出现未经用户同意收集个人信息的情况。

2. 初始化 X5 内核。
   - 进入课堂前，**必须先判断 X5 内核是否初始化完成，且初始化方法必须在主进程中执行**（SDK 提供了 TCICInitProvider.isMainProcess(context) 方法判断当前是否处于主进程）。
   - onViewInitFinished 方法已经回调，**初始化不允许重复调用，仅初始化1次即可。**

```
if (TCICInitProvider.isMainProcess(context)) {
    //初始化X5内核
    TCICManager.getInstance().initX5Core(licenseKey, new
TBSSdkManageCallback() {

        @Override
        public void onCoreInitFinished() {
        }

        @Override
        public void onViewInitFinished(boolean isX5Core) {
            //X5内核初始化完成,可以进课堂
        }
    });
}
```

> ⚠ **注意：**
> TCICManager.getInstance().initX5Core(licenseKey); 中的 licenseKey 参数需要通过 步骤六 提交工单联系我们获取 X5 内核的 licenseKey。

步骤八：获取进入课堂所需参数

TCICClassConfig 参数解释

1. 通过 控制台 进入**应用管理 > 应用配置**，获取 SDKAppId ，即为学校编号(schoolId)信息。
2. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(classId)信息。
3. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(userId)信息。
4. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 token 信息。
5. scene、lng、camera、mic、speaker为非必要参数，如果不设置则使用的是默认值。

字段	类型	必填	含义	备注
schoolId	int	是	学校编号	通过控制台进入**应用管理 > 应用配置**，获取 SDKAppId 。
classId	long	是	课堂编号	通过 CreateRoom 接口创建返回 RoomId 获取。
userId	string	是	用户账号	通过 RegisterUser 接口获取。
token	string	是	后台鉴权参数	通过 LoginUser 接口获取。
scene	string	否	场景名称	用于区分不同的定制布局，通过 SetAppCustomContent 接口配置。
lng	string	否	语言参数	当前支持中文(简体)、中文(繁体) 、English、韩语、日语、阿拉伯语、越南语、印尼语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi、id。 同时需要设置： TCICWebViewManager.getInstance().setClassLanguage(this, env, lng); lng参数。
camera	int	否	初始化开启摄像头	1为开启摄像头，0为关闭摄像头，默认 1。
mic	int	否	初始化开启麦克风	1为开启麦克风，0为关闭麦克风，默认 1。
speaker	int	否	初始化开启扬声器	1为开启扬声器，0为关闭扬声器，默认 1。

步骤九：调起组件主页面

只需传递 4 个参数就可调起 LCIC 组件主页面，分别为学校编号、课堂编号、用户账号和 token。

> **说明：**
> `schoolId` 同 SDKAppId 。

```
Intent intent = new Intent(getActivity(), TCICClassActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_SIN
GLE_TOP);
Bundle bundle = new Bundle();
TCICClassConfig initConfig = new TCICClassConfig.Builder()
    .schoolId(schoolId)
    .classId(classId)
    .userId(userId)
    .token(token)
    .build();
bundle.putParcelable(TCICConstants.KEY_INIT_CONFIG, initConfig);
intent.putExtras(bundle);
startActivity(intent);
```

如果您需要监听退出课堂通知，可以通过注册 TCICConstants.ON_CLASS_EXITED_ACTION 的本地广播，参见如下代码：

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

其它文档

- **我们建议您在使用 LCIC SDK 时，同时也接入腾讯 Bugly，帮助您快速发现并解决异常，同时掌握产品运营动态，及时跟进用户反馈。 接入指南参见 腾讯 Bugly 官网 。**
- **参见 开发 Demo 。**
- **自定义页面**

iOS
原生内核

最近更新时间：2025-12-09 14:43:52

开发环境要求

Xcode 14

前置准备

您已 注册腾讯云 账号，并完成 实名认证 。

步骤一：创建新的应用

1. 登录 实时互动-教育版 控制台 ，进入左侧导航栏的**概览**，选择**新建应用**。
2. 若尚未创建应用，则默认进入“创建应用”界面，输入应用名称，例如 TestLCIC。

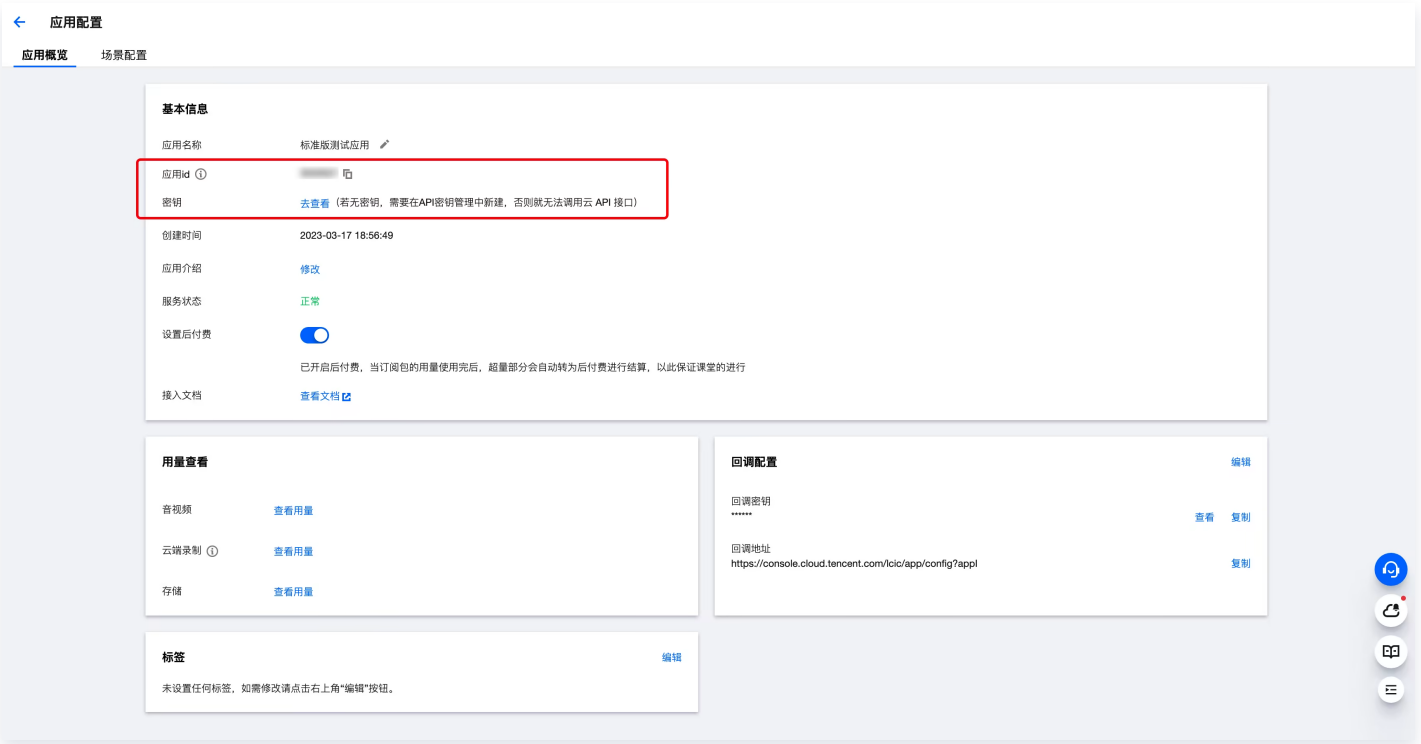
若您已创建应用，可前往**应用管理**中，选择使用已有应用即可。

说明：

- 移动端需要购买旗舰版或企业尊享版后，方可接入移动端。若需创建商用应用，可根据业务需求在 购买页 创建对应版本的应用。
- 应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入 应用管理 > **应用配置**，获取应用 ID（SDKAppId）。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中进行新建，具体可参考 访问密钥管理 。



步骤三：获取进入课堂所需参数

1. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(userId)信息。
2. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 token 信息。
3. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(classId)信息。

字段	类型	必填	含义	备注
userid	string	是	用户名。	通过 RegisterUser 接口获取。
classid	string	是	课堂 ID。	通过 CreateRoom 接口创建返回获取。
token	string	是	后台鉴权参数。	通过 LoginUser 接口获取。
role	string	否	进入课堂角色，默认空。	可选参数 supervisor(巡课/内容审查），只有已注册应用内巡课用户才有权限。
lng	string	否	语言参数，默认 zh-CN。	当前支持中文(简体)、中文(繁体) 、English、韩语、日语、阿拉伯语、越南语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi。

location	boolean	否	是否上报经纬度位置信息。	默认 false 不上报。
layout	string	否	页面布局。	默认顶部布局(top)，当前仅视频文档模式有效，支持双排布局（double）、右侧布局(right）、左侧布局(left)、三分布局(three)。
boardColor	string	否	白板颜色。	白板颜色设置，默认为：#182E25 ，支持 Hex 格式，也支持 rgba(0, 0, 0, .3)设置。
noEndClass	boolean	否	禁用下课。	助教进房时带上这个参数，在助教点击退出时，将隐藏「下课」，仅展示「离开」按钮。

快速跑通

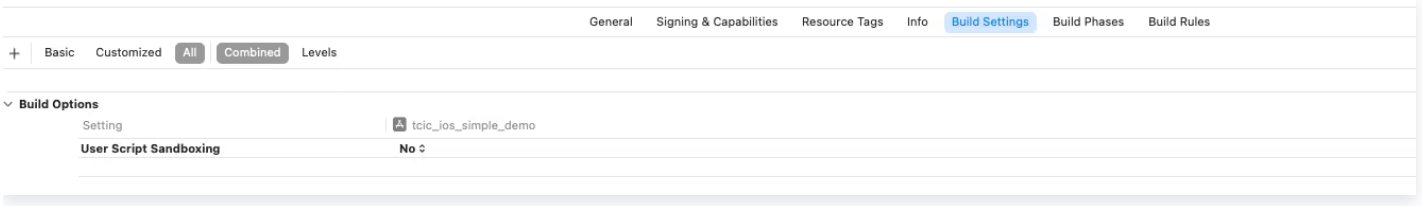
根据下面的步骤您可以快速跑通项目，具体代码也可以参见 Demo 。

步骤一：安装依赖

在您的 Podfile 中导入依赖文件。

```
// framework 使用静态集成
   use_frameworks! :linkage => :static
  // 根据不同的版本号，修改链接，下方示例为使用1.0.5 版本
  pod 'tcic_ios', :podspec =>
'https://ios.qcloudclass.com/1.0.9/tcic_ios.podspec'
```

如果没有 `Podfile` ，请使用 `pod init` 生成 `Podfile` 文件。
`User Script Sandboxing` 设置为 `No` 。



> **说明：**
> 请使用真机进行调试，音视频等插件在模拟器上无法看到效果。

步骤二：权限配置

在教学场景中，通常我们会用到音视频互动，屏幕共享等功能，因此需要应用配置对应的权限。
在 `Info.plist` 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

在您的页面首次进入的时候需要初始化 TCIC SDK，您可以根据您的实际业务来处理。

```
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

如果您需要监听课堂相关的事件，可添加相关回调，同初始化一样，您只需要添加一次即可。

```
@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");
    }
    TCICManager.shared.setCallback(callback);
```

步骤五：设置进入课堂的必要参数

您可以根据您的场景进入课堂页面，参数如下。

```
let headerConfig = TCICHeaderComponentConfig();
```

```
    let messageConfig = TCICMessageComponentConfig();
    headerConfig.headerLeftBuilder = headerLeftBuilder;
    let config = TCICConfig(
        token: "", /// 通过云API获取的token
        classId: "", /// 课堂id
        userId: "", /// 用户userId
        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， 具体参数
请参考TCICHeaderComponentConfig
        messageComponentConfig: messageConfig /// Message 组件配置，具体参
数请参考TCICMessageComponentConfig
        ...config, /// 您还可以设置其他组件的自定义配置
    );
    TCICManager.shared.setConfig(config)
```

步骤六：进入课堂页面

您可以根据业务需要在合适的时机打开课堂页面，如下展示点击按钮后，打开课堂页面。

```
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: "", /// 通过云API获取的token
        classId: "", /// 课堂id
        userId: "", /// 用户userId
        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， 具体参数
请参考TCICHeaderComponentConfig
        messageComponentConfig: messageConfig /// Message 组件配置，具体参
数请参考TCICMessageComponentConfig
```

```
            ...config, /// 您还可以设置其他组件的自定义配置
        );
        TCICManager.shared.setConfig(config)
        isActive = true
    }) {
        Text("打开TCIC页面")
            .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)
    }
```

步骤七：自定义 View

创建 Native View

```
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
    }
}
```

将 Native View 注册到 SDK

在设置课堂参数部分，可以传入`headerComponentConfig`，您只需要将您的自定义`native view` 传入即可。

```
    let headerConfig = TCICHeaderComponentConfig();

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

    headerConfig.headerLeftBuilder = headerLeftBuilder; // builder
    headerConfig.headerLeftBuilderWidth = 200; // 宽
    headerConfig.headerLeftBuilderHeight = 40 // 高

  let config = TCICConfig(
        token: "", /// 通过云API获取的token
        classId: "", /// 课堂id
        userId: "", /// 用户userId
        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， 具体参数
请参考TCICHeaderComponentConfig
        ...config, /// 您还可以设置其他组件的自定义配置
    );
    TCICManager.shared.setConfig(config)
```

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
    let isTestBackend: Bool          // 是否使用测试后端
}
```

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
}
```

callbacks

```
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
     * @param userId 用户信息
     */
    public var onMemberJoinedClassBlock: (([String: Any]) -> Void)?

    /**
     * 成员离开课堂回调
     * @param userId 用户信息
     */
    public var onMemberLeaveClassBlock: (([String: Any]) -> Void)?

    /**
     * 收到消息回调
     * @param messageJson 消息内容
     */
    public var onReceivedMessageBlock: (([String: Any]) -> Void)?

    /**
     * 发生错误回调
     * @param code 错误代码
     * @param message 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

// 设置头部视图
headerConfig.headerBuilder = {
    return MyHeaderView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
headerConfig.headerBuilderWidth = 300
headerConfig.headerBuilderHeight = 50

// 设置头部操作视图
headerConfig.headerActionsBuilder = {
    return MyHeaderActionsView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
headerConfig.headerActionsBuilderWidth = 150
headerConfig.headerActionsBuilderHeight = 40

// 设置右侧视图
headerConfig.headerRightBuilder = {
    return MyHeaderRightView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
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 = {
    return MyMessageHeaderView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
messageConfig.messageHeaderBuilderWidth = 200
messageConfig.messageHeaderBuilderHeight = 30

// 设置消息气泡视图
messageConfig.messageBubbleBuilder = {
    return MyMessageBubbleView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
messageConfig.messageBubbleBuilderWidth = 250
messageConfig.messageBubbleBuilderHeight = 80

// 设置消息项视图
messageConfig.messageItemBuilder = {
    return MyMessageItemView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
messageConfig.messageItemBuilderWidth = 300
messageConfig.messageItemBuilderHeight = 60

// 设置消息行视图
messageConfig.messageRowBuilder = {
    return MyMessageRowView(messenger:
TCICManager.shared.flutterEngine.binaryMessenger)
}
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
```

技术支持

如果您在使用过程中遇到问题，可以通过以下方式获取技术支持：

- 腾讯云实时互动-教育版官方文档
- 腾讯云开发者社区
- GitHub Issues

Web 内核

最近更新时间：2025-12-09 14:43:52

开发环境要求

Xcode 14

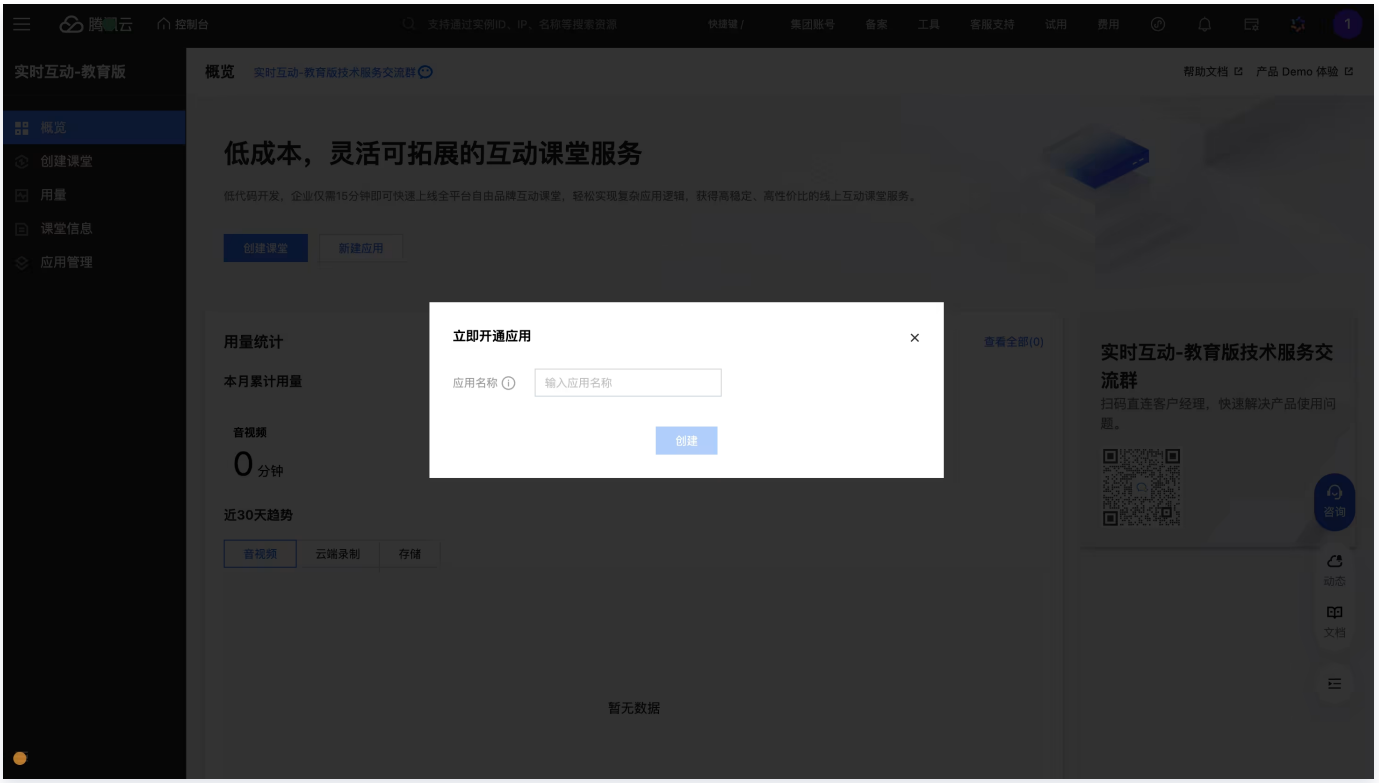
前提条件

您已 注册腾讯云 账号，并完成 实名认证 。

操作步骤

步骤一：创建新的应用

1. 登录 实时互动-教育版 控制台 ，进入左侧导航栏的**概览**，选择**新建应用**。
2. 若尚未创建应用，则默认进入“创建应用”界面，输入应用名称，例如 TestLCIC。



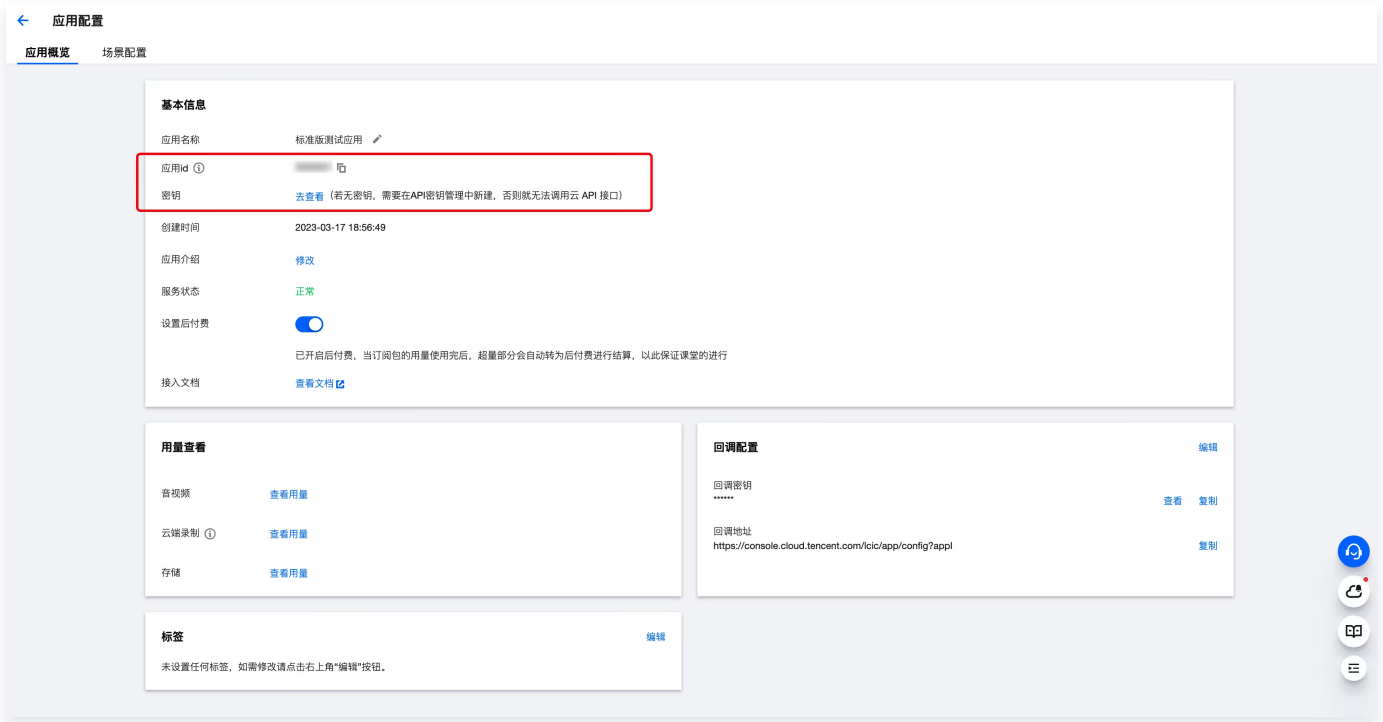
若您已创建应用，可前往**应用管理**中，选择使用已有应用即可。

说明：

- 移动端需要购买旗舰版或企业尊享版后，方可接入移动端。若需创建商用应用，可根据业务需求在购买页 创建对应版本的应用。
- 应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入**应用管理 > 应用配置**，获取 SDKAppId 。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中进行新建，具体可参考 访问密钥管理 。



步骤三：导入 SDK

pod 集成 SDK

LCIC SDK 已经发布到 cocoapods 库，您可以通过配置 podfile 下载安装。

```
pod 'TCICSDK_Pro', '1.8.5.17'
```

```
pod 'TXLiteAVSDK_Professional', '12.7.19324'
```

步骤四：配置 App 权限

在主 App 的 info.plist 中配置 App 的权限，LCIC SDK 需要以下权限：

```
<key>NSCameraUsageDescription</key>
<key>NSMicrophoneUsageDescription</key>
<key>NSPhotoLibraryAddUsageDescription</key>
<key>NSPhotoLibraryUsageDescription</key>
```

步骤五：环境参数设置

在启动课堂前可设置相关环境参数，如语言、域名等参数，设置如下：

```
// 设置课堂域名，默认域名为：class.qcloudclass.com
// targetDomain 必须为腾讯提供域名
[TCICClassController setDomain:targetDomain];
// 设置H5端版本号，默认为："latest"。一般情况不需要调
[TCICClassController setH5Version:targetVersion];
// 设置课堂语言类型，默认为中文："zh"
[TCICClassController setClassLanuage:[msgDic objectForKey:@"lng"]];
// 预加载逻辑需要放在最后，前面的调用都会清除预加载内容
[TCICClassController preloadClass];
```

步骤六：获取进入课堂所需参数

1. 通过 控制台 进入**应用管理 > 应用配置**，获取 SDKAppId ，即为学校编号(schoolId)信息。
2. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(classid)信息。
3. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(userid)信息。
4. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 token 信息。
5. scene、lng、camera、mic、speaker 为非必要参数，如果不设置则使用的是默认值。

字段	类型	含义	备注	必填
schoolId	int	学校编号	通过控制台进入**应用管理 > 应用配置**，获取 SDKAppId 。	是
classId	long	课堂编号	通过 CreateRoom 接口创建返回 RoomId 获取。	是

userId	string	用户账号	通过 RegisterUser 接口获取。	是
token	string	后台鉴权参数	通过 LoginUser 接口获取。	是
scene	string	场景名称	用于区分不同的定制布局，通过 SetAppCustomContent 接口配置 [roomConfig setValue:@"scene_name" forKey:@"scene"];	否
lng	string	语言参数	当前支持中文(简体)、中文(繁体)、English、韩语、日语、阿拉伯语、越南语、印尼语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi、id。可通过此接口设置 [roomConfig setValue:@"en" forKey:@"language"]。	否
camera	int	初始化开启摄像头	1为开启摄像头，0为关闭摄像头，可通过 roomConfig.jsParams 设置。	否
mic	int	初始化开启麦克风	1为开启麦克风，0为关闭麦克风，可通过 roomConfig.jsParams 设置。	否
speaker	int	初始化开启扬声器	1为开启扬声器，0为关闭扬声器，可通过 roomConfig.jsParams 设置。	否

步骤七：调起组件主页面

只需传递 4 个参数就可调起 LCIC 组件主页面，分别为学校编号、课堂编号、用户账号和 token。

```
TCICClassConfig *roomConfig = [[TCICClassConfig alloc] init];
roomConfig.schoolId = 123456;
roomConfig.userId = "test";
roomConfig.token = "test_token";
roomConfig.classId = 654321;
[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");
}
```

如果您需要监听退出课堂通知，可以通过注册 TCICExitClassRoomCompleteNotify 的本地广播，参见如下代码：

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(yourselector)

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

步骤八：SDK 授权申请

需要您提交 腾讯云工单，向我们发送 SDK 权限申请。请按以下模板提供对应信息。在信息确认无误的情况下，我们将会在1个工作日完成。

> **注意：**
> - 一个旗舰版仅支持授权一个正式包名，请确认无误后发送相关信息。
> - 包名用于快直播播放器签名授权，请提供所需授权的正式应用的 App Name、Package Name 和 Bundle ID 信息。

分类	说明
问题标题	实时互动-教育版 iOS SDK 授权申请
问题主要内容	公司名称。如 xxx 有限公司。
	个人姓名
	联系方式
	App Name
	Package Name（Android）

	Bundle ID（iOS）

高级功能

移动端屏幕分享

1. 创建 `App Group`，参见 TRTC 官网文档 > 步骤 1：创建 App Group。
2. 创建 `Broadcast Upload Extension`，参见 TRTC 官网文档 > 步骤 2：创建 Broadcast Upload Extension。
3. 为新创建的 `Target`，依赖 `TCICSDK_ReplayKit`，如下。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_Pro_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 {
    if (kSupportSceenShare) {
        switch (sampleBufferType) {
            case RPSampleBufferTypeVideo:
            case RPSampleBufferTypeAudioApp:
```

```
                [[TCICScreenKit sharedScreenKit]
processSampleBuffer:sampleBuffer withType:sampleBufferType];

                [[TXReplayKitExt sharedInstance]
sendSampleBuffer:sampleBuffer withType:sampleBufferType];
                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 即可，且在进入课堂前，设置 AppGroup 即可。

> **说明：**
> `schoolId` 同 SDKAppId 。

```
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

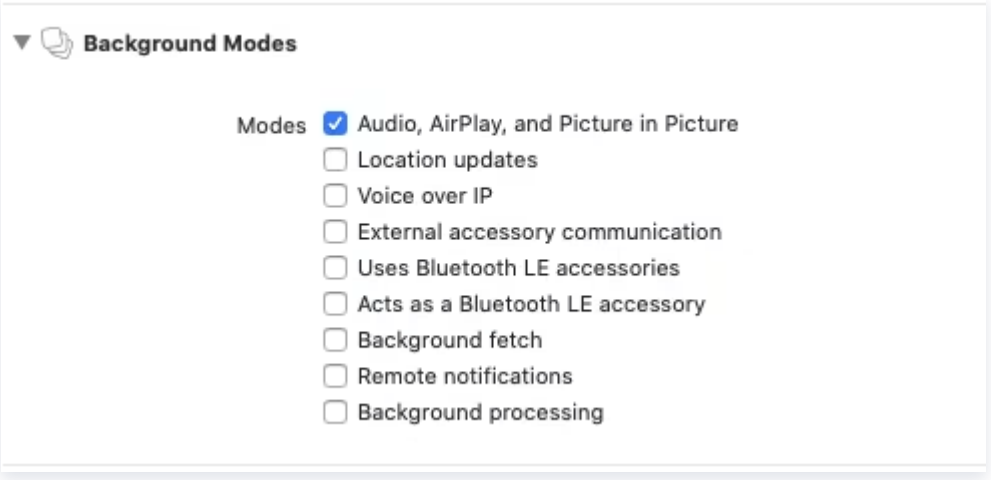
1. TCICSDK 中已支持屏幕分享的触发按钮，具体可参见 TRTC 官网文档 > 步骤 4：增加屏幕分享的触发按钮（可选），但该功能有限制条件。

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：

长按“录屏” 按钮　　点击“开始直播”

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` 。



其它文档

- 我们建议您在使用 LCICSDK 时，同时也接入腾讯 Bugly，帮助您快速发现并解决异常，同时掌握产品运营动态，及时跟进用户反馈。 接入指南参见 腾讯 Bugly 官网 。
- 参见 开发 Demo 。

- 自定义页面

Windows 和 macOS

最近更新时间：2025-11-11 15:37:01

为提供跨平台的桌面应用程序，Windows 和 macOS 端使用了 Electron 的方案进行开发，本文主要介绍如何快速将腾讯云 LCIC Electron SDK 集成到您的项目中。

> **说明：**
> 由于虚拟背景功能是与应用程序名称、包名一一对应绑定，如不是直接使用实时互动-教育版下载的 Demo 或域名，则需要联系商务申请购买 虚拟背景能力 。

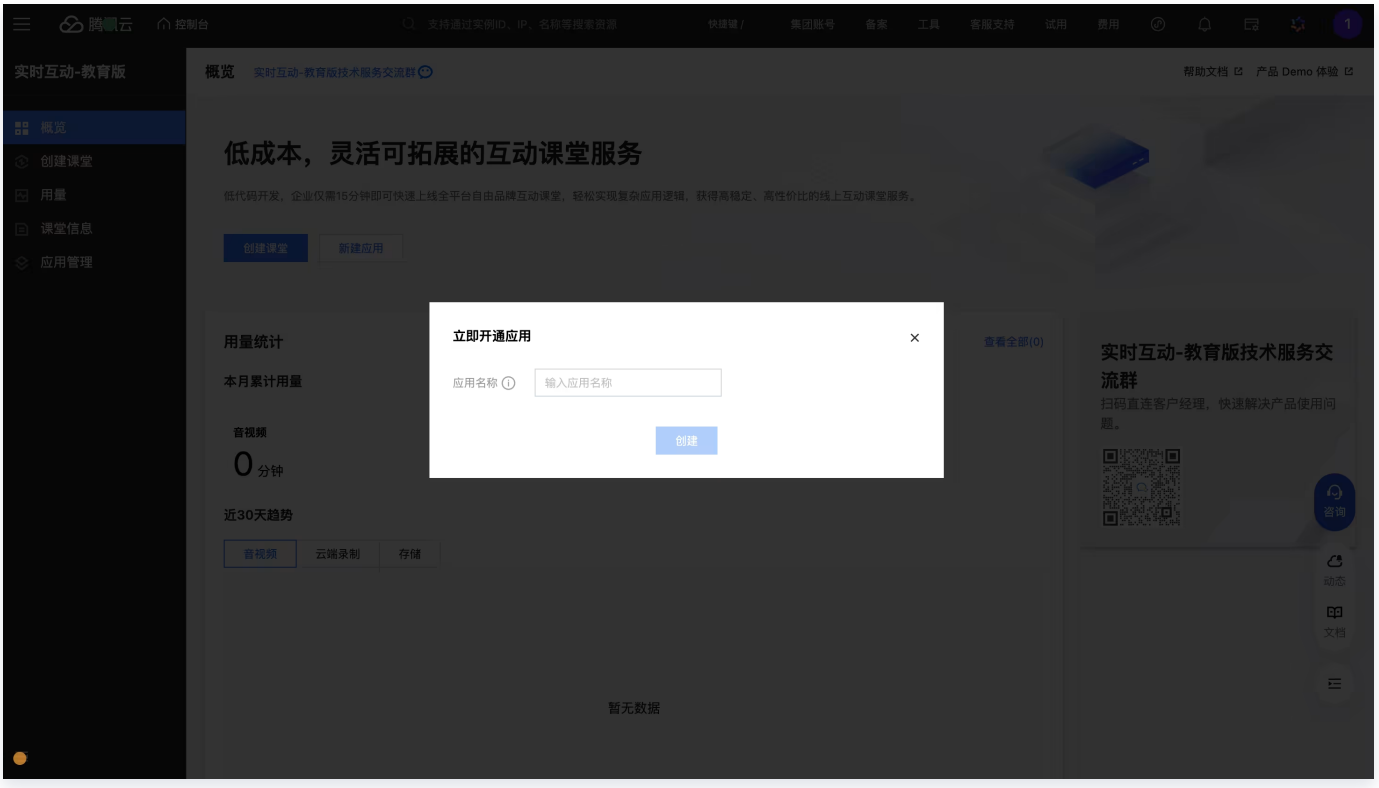
前提条件

您已经 注册腾讯云 账号，并完成 实名认证 。

操作步骤

步骤一：创建新的应用

1. 登录 实时互动-教育版控制台 ，进入左侧导航栏的**概览**，选择**新建应用**。
2. 若尚未创建应用，则默认进入“创建应用”界面，输入应用名称，例如 TestLCIC。



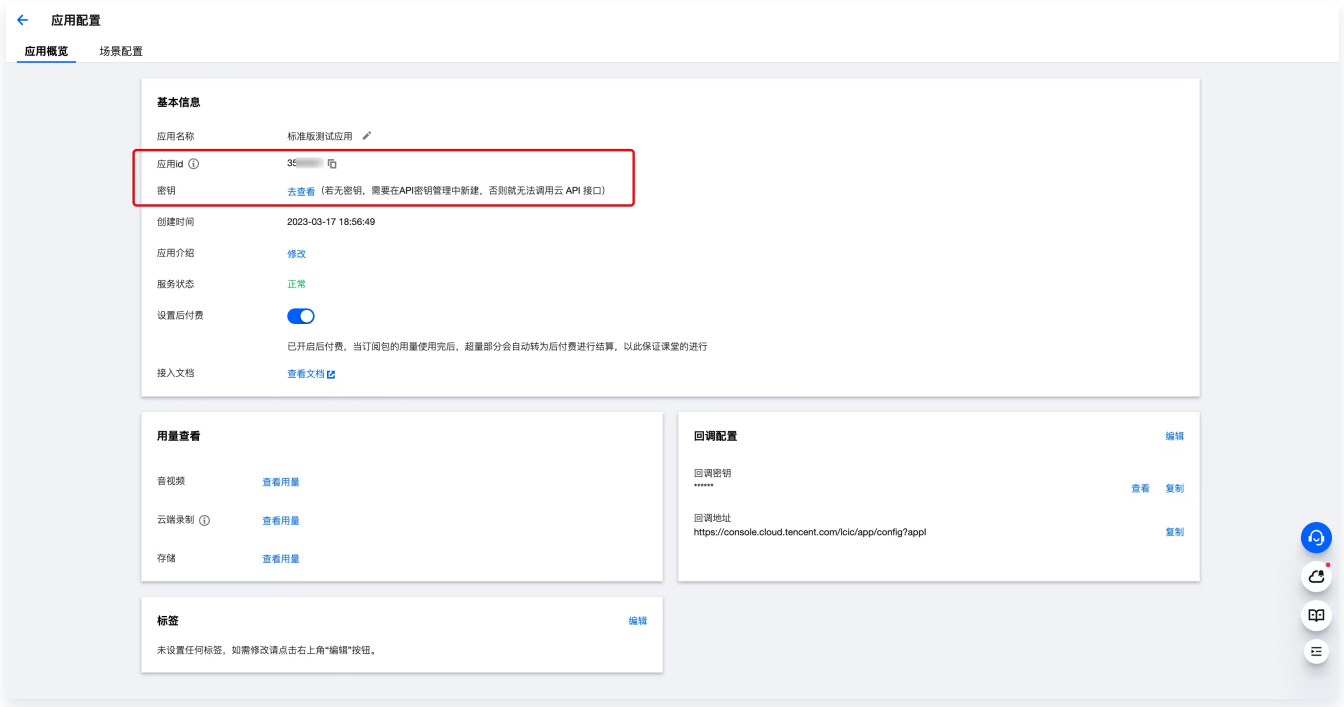
若您已创建应用，可前往应用管理中，选择使用已有应用即可。

说明：

- 每个账号可免费领用一个试用版应用，若需创建商用应用，可根据业务需求在 购买页 创建对应版本的应用。
- 应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入 应用管理 > **应用配置**，获取应用 id（SDKAppId）。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中进行新建，具体可参见 访问密钥管理 。



步骤三：获取进入课堂所需参数

1. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(`userid`)信息。
2. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 `token` 信息。
3. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(`classid`)信息。
4. 选择需要集成的 课堂版本 ，一般业务侧集成最新版本即可。

5. 其中 `scene` 、 `debugjs` 、 `debugcss` 为非必填参数，在需要自定义 UI 时才需设置，具体可参考 自定义 UI 集成 。其中 `debugjs` 和 `debugcss` 只用于自定义布局、组件时的调试，且只支持通过 `localhost` 或 `127.0.0.1` 的地址进行访问，在发布阶段请勿使用此参数。
6. `lng` 、 `location` 、 `layout` 也是非必填参数，业务侧可自行判断是否需要传入，不传则使用默认值，其中 `layout` 参数只有在教室布局为视频+文档布局(`videodoc`)时才生效。

字段	类型	必填	含义	备注
userid	string	是	用户名	通过 RegisterUser 接口获取
classid	string	是	课堂 ID	通过 CreateRoom 接口创建返回获取
token	string	是	后台鉴权参数	通过 LoginUser 接口获取
version	string	否	课堂版本号	（从 tcic-electron-sdk v1.9.0版本起将正式废弃此参数）通过发布日志选择对应版本
scene	string	否	场景名称	区分不同的定制布局，通过 SetAppCustomContent 接口配置
role	string	否	进入课堂角色，默认空	可选参数 supervisor(巡课/内容审查) 只有已注册应用内巡课用户才有权限
debugjs	string	否	自定义 UI 的 JS 链接	通过自定义 UI 集成方式获取
debugcss	string	否	自定义 UI 的 CSS 链接	通过自定义 UI 集成方式获取
lng	string	否	语言参数，默认 zh-CN	当前支持中文(简体)、中文(繁体)、English、韩语、日语、阿拉伯语、越南语、印尼语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi、id。
location	boolean	否	是否上报经纬度位置信息	默认 false 不上报
layout	string	否	页面布局	默认顶部布局(top)，当前支持双排布局(double)、右侧布局(right)、左侧布局

				(left)、三分布局(three)

步骤四：进入课堂

针对不同业务场景需要，我们提供了以下两种接入集成方式。

方式一：URL 拼接

通过浏览器呼起客户端的能力。客户端分为腾讯云提供的 LCIC-Demo 客户端和自定义客户端。区别在于客户端图标、名称、初始页面不一样。

使用浏览器直接呼起客户端

1. 下载客户端并安装。
   - Windows 64位
   - Windows 32位
   - macOS
2. 通过链接唤起客户端，当用户在浏览器中访问或跳转到 URL `tcic://class.qcloudclass.com/latest/class.html?classid=${classId}&userid=${userId}&token=${token}` 时，浏览器会请求打开 LCIC-Demo 客户端。

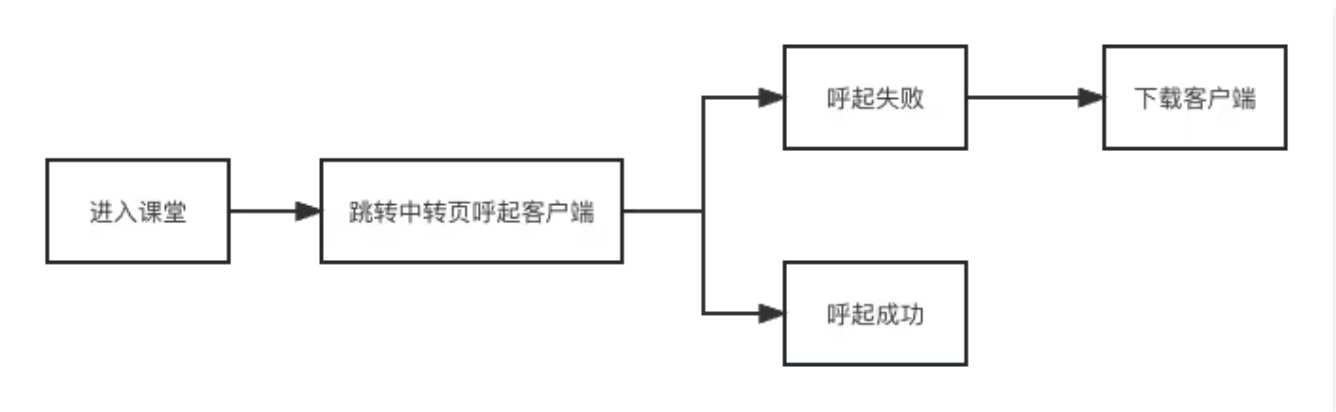
中转页呼起方式（推荐）

通过简单快速集成我们的 LCIC Electron SDK，业务侧实现跳转中转页，识别成功就直接跳转，失败则优先下载客户端。如果用户取消或者不下载，再通过跳转打开 Web 版本的上课页面。

具体流程：

在进入课堂时需先跳转至一个中转页，在中转页内处理用户跳转逻辑。获取上课参数拼接到此 URL `tcic://class.qcloudclass.com/latest/class.html?classid=${classId}&userid=${userId}&token=${token}` 上，即可唤起客户端应用。

流程可参见下图：



下载 ElectronProtocolCheck.js 引入自己的项目中。中转页中的示例代码如下：

```
// ElectronProtocolCheck 文件代码见github示例
```

```
import ElectronProtocolCheck from './ElectronProtocolCheck';

// 需要进入课堂的id
const classId  = 368507569;
// 当前进入课堂用户的id
const userId = "JIUzI1NiIsIn123456";
// token 需要动态从后台接口获取，防止登录态过期失效
const token =
 'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwNzQwMjEsImlhdCI6MT
Y3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JYWpleVh2RWwiLCJzY2
hvb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCNVhCczZHT1BnVXpweU
gifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII';
const url = `tcic://class.qcloudclass.com/latest/class.html?
classid=${classId}&userid=${userId}&token=${token}`;
console.log(`callClient->start: ${url}`);
// 唤起客户端
ElectronProtocolCheck(
  url,
  () => {
    // 呼起成功
    console.log('callClient->success! ');
  },
  () => {
    console.log('callClient->failed! ');
    // 没有呼起来，建议此处实现提示下载的弹窗

    // 如果用户点击下载 --> 换弹窗口信息，打开客户端。

    // 如果用户取消下载或关闭弹窗。在让用户尝试体验web/H5 版本的课堂
    // 也可以加上超时弹窗 一般 2500ms 内用户没有做出点击操作，可以出现模态点击框
  },
  () => {
      // 浏览器不支持等情况，可以走 web 链接加载
  });
```

说明：
在拼接的 URL 中，若传入的 `userid` 与当前课堂指定的 `teacherid` 是一致的，则当前用户为老师。若与当前课堂的助教 ID(`assistantid`)一致，则为助教，否则为学生。

方式二：SDK 集成

我们提供了一个 electron demo 供您参考，集成方式如下：

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

3. 在项目脚本里引入模块后，调用初始化接口并传入之前获取的参数调起课中页面。

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

TCIC.initialize({
  classId: '368507569',
  userId: '123456',
  token:
'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwNzQwMjEsImlhdCI6M
TY3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JYWpleVh2RWwiLCJ
zY2hvb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCNVhCczZHT1BnV
XpweUgifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII',
  // 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?: Record<string, string>; // 自定义参数，会原样传给web端
  onReady() {
    // 可选，拉起课堂窗口时的回调
  },
  onClose() {
    // 可选，所有窗口都被关闭时的回调
  },
  sign: string, // 可选，客户业务使用用户签名
  cid: string, // 可选，客户业务使用用户id
```

```
  uid: string, // 可选，客户业务使用用户id
  // sdk 初始化成功后的回调
  afterInit?: () => void;
})
```

高级功能

流水线打包接入

若您希望生成的客户端拥有自己的品牌 logo 和应用名的同时，能快速集成现有业务系统，可以通过使用我们提供的流水线打包方式进行集成。
具体流程参见下图：

1. **处理业务逻辑**
   - **在打包后，业务侧可通过 `window` 全局变量上的 `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:
'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwNzQwMjEsImlhd
CI6MTY3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JYWpleVh
2RWwiLCJzY2hvb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCN
VhCczZHT1BnVXpweUgifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII'
,
};
// 点击进入课堂的逻辑方法
window.joinClass(options);

// 如何同时兼容原有逻辑呢？请参考下面的代码逻辑

// 判断是否存在这个方法
if (window.joinClass) {
  window.joinClass(options);
} else {
```

```
    // 原有的进入课堂逻辑
}
```

- 业务侧需通过 `window` 全局变量上的 `closeWin` 关闭当前的客户端。

```
window.closeWin();
```

2. 准备对应的物料清单，具体如下所示：

字段	含义	必填
AppName	App 名称，如XX课堂	是
Logo	应用 Logo，请提供规格为 256x256的 ico/png 图片	是
URL	业务 URL	是

> **注意：**
> 业务 URL 应是已经具备了登录态的完整 URL，如果没有登录态，能自动跳转至登录入口。配置的业务 URL 不应是登录页，否则用户每次打开都需要重新登录。

3. 发送打包申请邮件，请按照以下格式进行发送，并以附件的形式附带上述物料，信息及物料确认无误的情况下我们将会在1个工作日完成打包。

> **说明：**
> 收件人：请联系对接您的腾讯云商务经理或产品经理获取
> 主题
> 申请客户端打包
> 内容
> 公司名称：xxx 有限公司
> 个人姓名：
> 联系方式：
> 打包物料（附件）

自定义 UI 集成

为满足不同客户需求，LCIC Electron 目前还提供了自定义 UI 的集成方案。用户可自定义业务侧课中的布局及样式，通过 自定义 UI 部分 可以获取到业务侧的JS 及 CSS 链接，将 `debugjs` 及 `debugcss` 参数拼接到上方的链接上即可(**此参数只用于调试**)，如下代码所示：

```
// URL拼接方式
const url = `tcic://class.qcloudclass.com/latest/class.html?
classid=${classId}&userid=${userId}&token=${token}debugjs=http://localho
st:443/demo/dist/myLib.umd.min.js&debugcss=http://localhost:443/demo/dis
t/myLib.css`;

// SDK集成方式
TCIC.initialize({
  classId: '368507569',
  userId: '123456',
  token:
'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwNzQwMjEsImlhdCI6MTY
3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JYWpleVh2RWwiLCJzY2h
vb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCNVhCczZHT1BnVXpweUg
ifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII',
  debugjs: 'http://localhost:443/demo/dist/myLib.umd.min.js',
  debugcss: 'http://localhost:443/demo/dist/myLib.css',
})

// 流水线打包方式
const options = {
  classId: '368507569',
  userId: '123456',
  token:
'yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODAwNzQwMjEsImlhdCI6MTY
3OTQ2OTIyMSwiaXNzIjoibFpNQ2tvTjNkSGlnVmhhcXJkdFc0cU9JYWpleVh2RWwiLCJzY2h
vb2xfaWQiOjM5MjMxOTMsInVzZXJfaWQiOiIyTG9XREU2aHhzOUNCNVhCczZHT1BnVXpweUg
ifQ.2wzh6eUC4llbbGhchGDOYbDrsdSdymfP3zjLLPjnOII',
  debugjs: 'http://localhost:443/demo/dist/myLib.umd.min.js',
  debugcss: 'http://localhost:443/demo/dist/myLib.css',
};
window.joinClass(options);
```

当自定义 JS 与 CSS 调试完成后，可通过云 API 接口 SetAppCustomContent 或 **控制台 > 应用配置 > 场景配置**将场景与自定义的 JS、CSS 链接进行绑定，在进入课堂时将 `scene` 参数拼接到 URL 或添加到对应入参上，即可加载对应场景的布局及组件。在涉及多种班型、多种布局时，业务侧可根据此参数实现场景的切换。

自定义业务域名

在课中页面时，若业务侧想隐藏课堂域名，只显示业务域名，可通过 内容分发网络控制台(CDN) 新建业务域名，并回源到课堂域名即可，详细流程请参见 自定义业务域名 。

其他相关文档

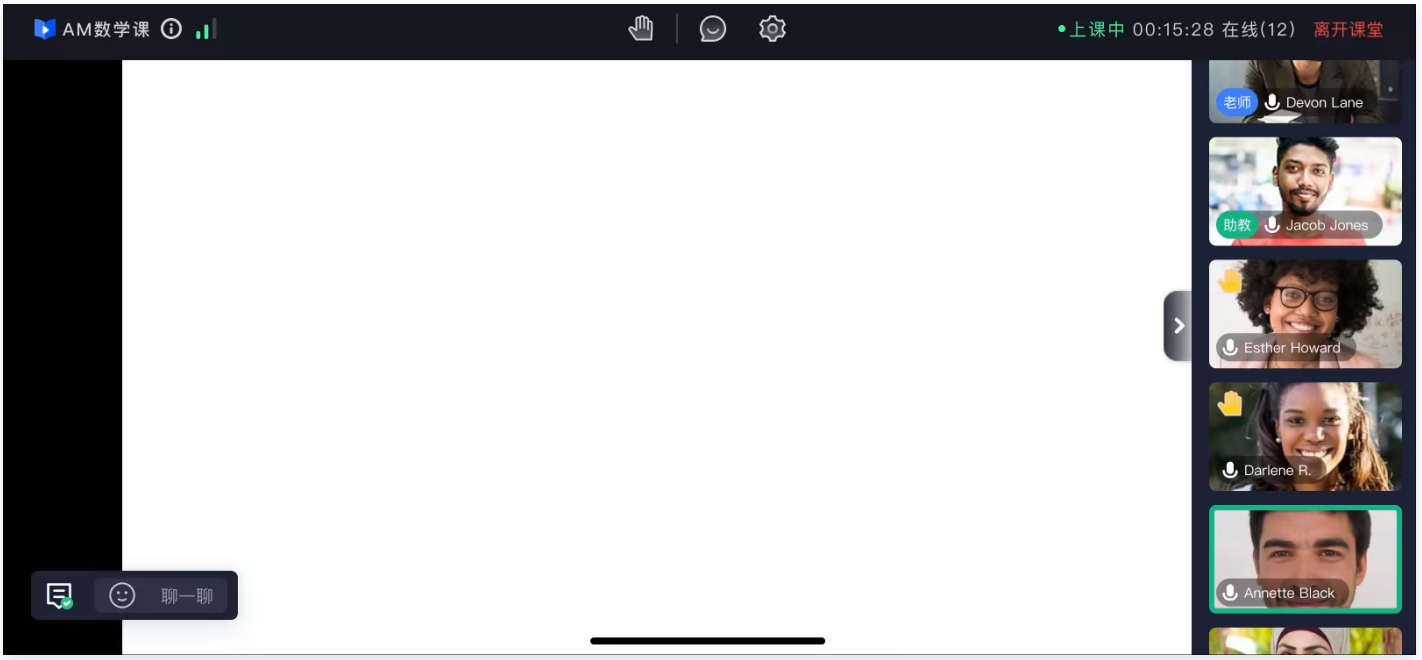
- LCIC API
- 自定义 UI 集成

Flutter

最近更新时间：2025-09-15 17:42:02

腾讯云实时互动-教育版（原低代码互动课堂 LCIC）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。在教育、医疗、金融和企业培训等领域，可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、音视频连麦等功能。**



支持平台

- Android (API 21+)
- iOS (12.0+)

环境要求

Flutter 环境

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

Android 环境

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

- Kotlin：支持

iOS 环境

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

在线 Demo

单击查看 Demo 地址 。

前置准备

在使用 `TCIC Client` 前，您需要先完成以下前置准备。

步骤一：创建新的应用

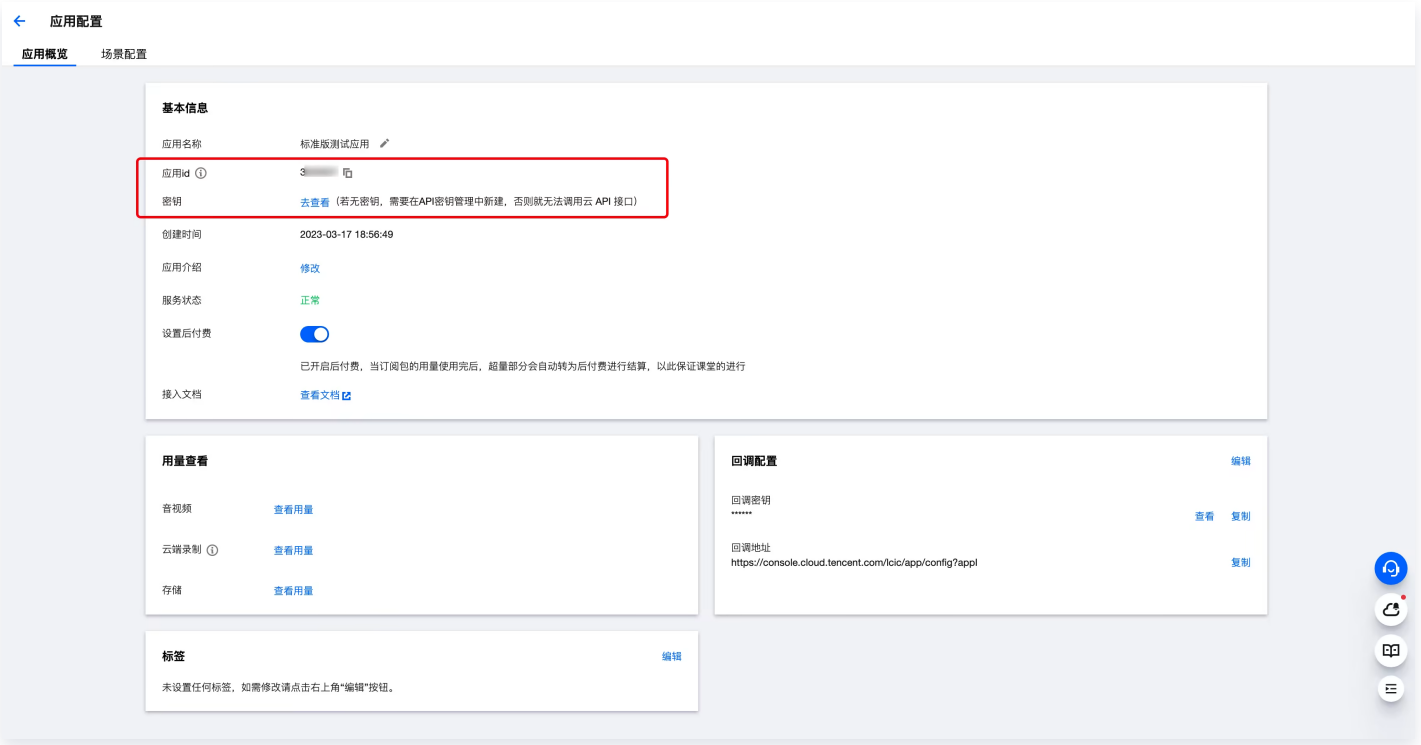
1. 登录 实时互动-教育版控制台 ，进入左侧导航栏的概览，单击**新建应用**。
2. 若尚未创建应用，则默认进入"创建应用"界面，输入应用名称，例如 TestLCIC。

若您已创建应用，可前往 应用管理 中，选择使用已有应用即可。

> **说明：**
> 每个账号可免费领用一个试用版应用，若需创建商用应用，可根据业务需求在 购买页 创建对应版本的应用。应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入 应用管理 > 应用配置，获取应用 ID（SDKAppId）。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中新建，具体可参见 访问密钥管理 。



步骤三：获取进入课堂所需参数

1. 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(userid)信息。
2. 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 token 信息。
3. 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(classid)信息。

字段	类型	必填	含义	备注
userid	string	是	用户名。	通过 RegisterUser 接口获取。
classid	string	是	课堂 ID。	通过 CreateRoom 接口创建返回获取。
token	string	是	后台鉴权参数。	通过 LoginUser 接口获取。
role	string	否	进入课堂角色，默认空。	可选参数 supervisor(巡课/内容审查)，只有已注册应用内巡课用户才有权限。
lng	string	否	语言参数，默认 zh-CN。	当前支持中文(简体)、中文(繁体)、English、韩语、日语、阿拉伯语、越南语。可拼接相应参数，展示对应语

				种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi。
location	boolean	否	是否上报经纬度位置信息。	默认 false 不上报。
layout	string	否	页面布局。	默认顶部布局(top)，当前仅视频文档模式有效，支持双排布局（double）、右侧布局(right）、左侧布局(left)、三分布局(three)。
boardColor	string	否	白板颜色。	白板颜色设置，默认为：#182E25，支持 Hex 格式，也支持 rgba(0, 0, 0, .3)设置。
noEndClass	boolean	否	禁用下课。	助教进房时带上这个参数，在助教点击退出时，将隐藏「下课」，仅展示「离开」按钮。

快速跑通

安装依赖

在您的 Flutter 项目中执行 `flutter pub add tcic_client_ui` 安装依赖。

权限相关配置

在教学场景中，通常我们会用到音视频互动，屏幕共享等功能，因此需要应用配置对应的权限。

Android 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 `android/app/src/main/AndroidManifest.xml` 中添加屏幕录制辅助 Activity：

```
<activity

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

3. build.gradle 配置

在 `android/app/build.gradle.kts` 中配置：

1. `ndkVersion` 版本设置为 `27.0.12077973`。
2. 配置 `packagingOptions` 解决 so 文件冲突问题。

```
android {
    namespace = "com.tencent.tcic"
    compileSdk = flutter.compileSdkVersion
    ndkVersion = "27.0.12077973"

    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 配置

1. 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>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>
```

跑通代码

配置完权限相关后，在前置条件中您了创建应用并且已经获取到了必要的 `classId`，`userId`，`token` 参数，只要将这些参数传入到组件中使用即可。

```
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';

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

// 配置参数
final config = TCICConfig(
  userId: 'user123',  /// 前置条件步骤中注册用户使用的用户id
  classId: 'class456', /// 前置条件步骤中创建的课堂id
  role: RoleEnum.student,
  token: 'your_token_here', /// 前置条件步骤中登录用户获取的token
```

```
  // 可选参数
  isTestBackend: false, /// 是否使用后台测试环境
);

// 根据您的业务场景，跳转到课堂页面
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => TCICView(
      controller: controller,
      config: config,
    ),
  ),
);
```

通过上述步骤您已经成功集成了课堂组件，您可以在您的应用中使用该组件来实现课堂功能。

跨应用屏幕共享

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的时候创建，并负责接收系统采集到屏幕图像。因此需要：

1. 创建 App Group，并在 Xcode 中进行配置（可选）。这一步的目的是让 Extension 录屏进程可以同主 App 进程进行跨进程通信。
2. 在您的工程中，新建一个 Broadcast Upload Extension 的 Target，并在 GitHub 中获取专门为扩展模块定制的 TXLiteAVSDK_ReplayKitExt.framework 。

> **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。因此，尽管步骤较多，建议尽量配置正确的 App Group 以保障屏幕分享功能的稳定性。

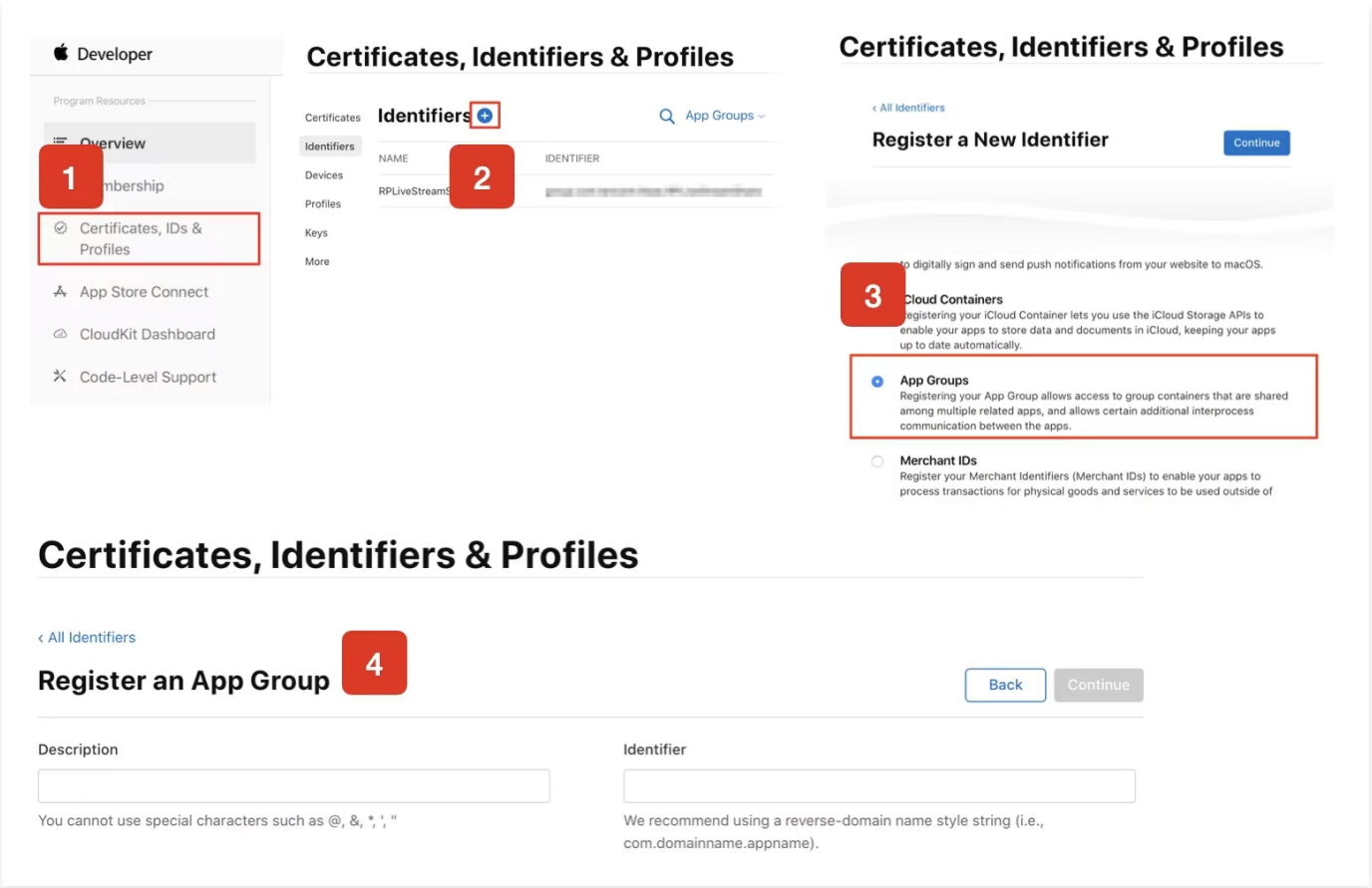
步骤1：创建 App Group

使用您的账号登录 Apple Developer，按以下步骤操作。

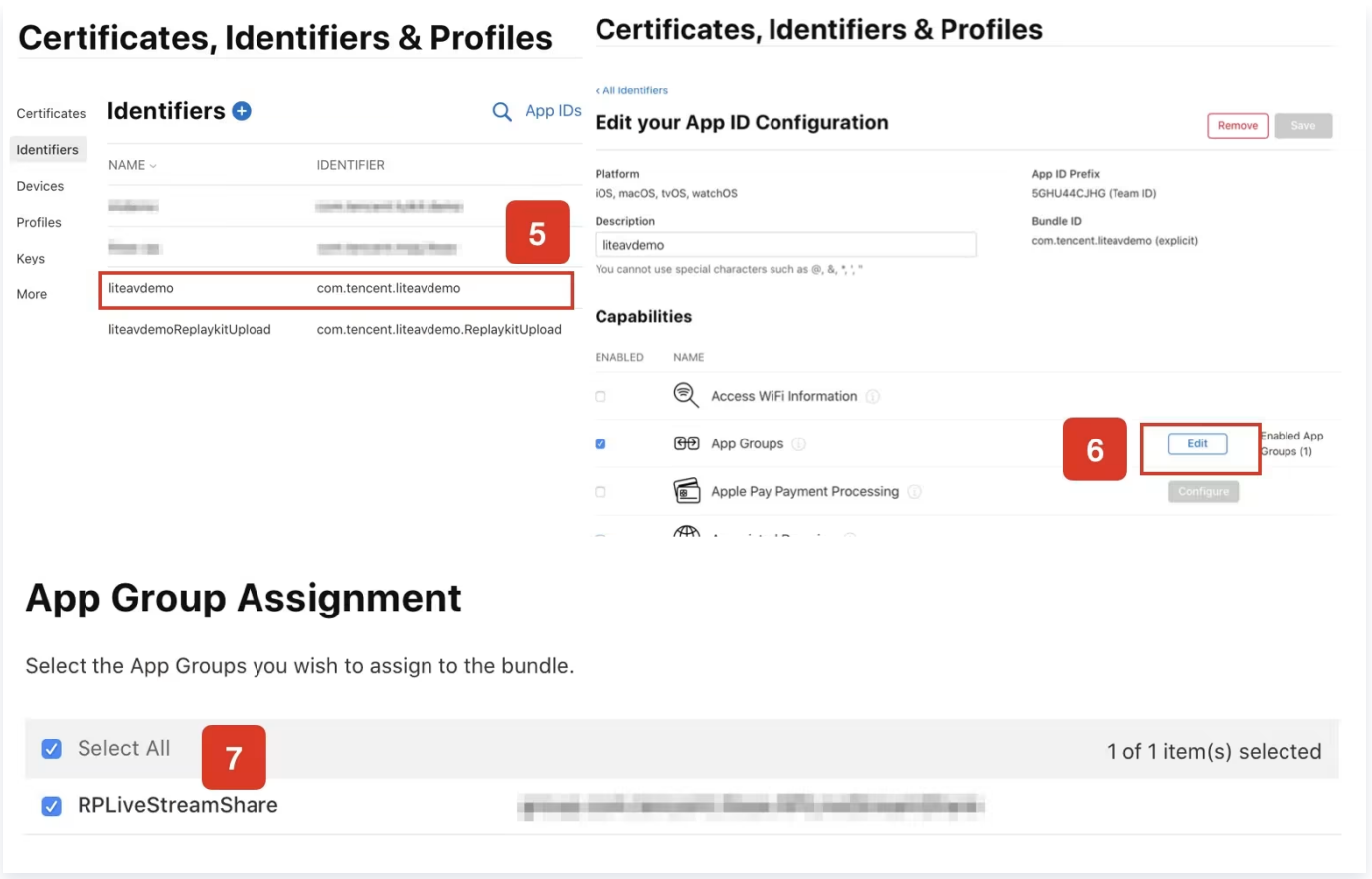
> **注意：**
> 完成后需要重新下载对应的 Provisioning Profile。

1. 单击左侧的 **Certificates, IDs & Profiles**。
2. 在右侧的界面中单击加号。
3. 选择 **App Groups**，单击 **Continue**。

4. 在弹出的表单中填写 Description 和 Identifier，其中 Identifier 需要传入接口中对应的 AppGroup 参数。完成后单击 **Continue**。



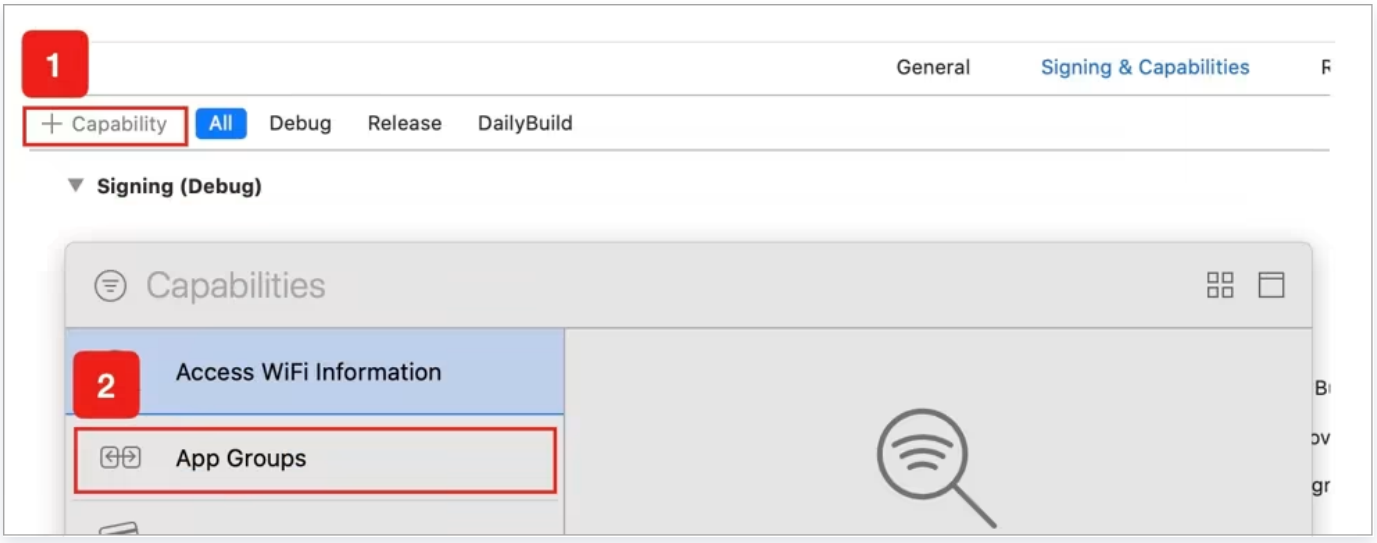
5. 回到 Identifier 页面，在上边的菜单中选择 App IDs，然后单击您的 App ID（主 App 与 Extension 的 AppID 需要进行同样的配置）。
6. 选中 **App Groups** 并单击 **Edit**。
7. 在弹出的表单中选择您之前创建的 App Group，单击 **Continue** 返回编辑页。然后，单击 **Save** 保存。



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

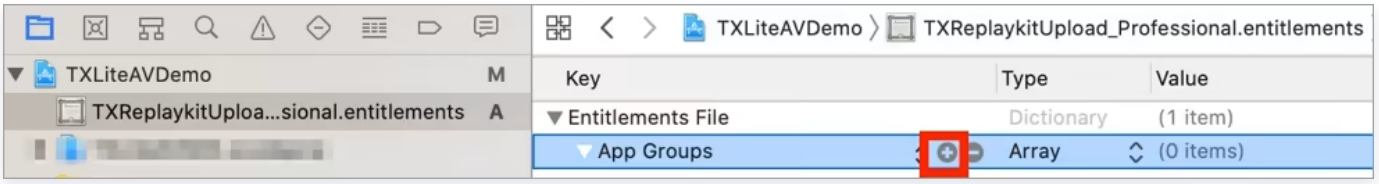
步骤2：创建 Broadcast Upload Extension

1. 在 Xcode 菜单依次单击 **File** > **New** > **Target...**，选择 **Broadcast Upload Extension**。
2. 在弹出的对话框中填写相关信息，不用勾选 Include UI Extension，单击 **Finish** 完成创建。
3. 将下载到的 SDK 压缩包中的 TXLiteAVSDK_ReplayKitExt.framework 拖动到工程中，勾选刚创建的 Target。
4. 选中新增加的 Target，依次单击 **+ Capability**，双击 **App Groups**，如下图：



AddCapability

操作完成后，会在文件列表中生成一个名为 Target.entitlements 的文件。如下图所示，选中该文件并单击 + 号填写上述步骤中的 App Group 即可。



AddGroup

5. 选中主 App 的 Target ，并按照上述步骤对主 App 的 Target 做同样的处理。
6. 在新创建的 Target 中，Xcode 会自动创建一个名为 "SampleHandler.swift" 的文件，用如下代码进行替换。需将代码中的 APPGROUP 改为上文创建的 App Group Identifier。

```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt

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]?) {
        // 用户已请求开始直播。UI 扩展程序的设置信息可以通过ut optional.
        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() {
        // 用户已请求结束广播。
        TXReplayKitExt.sharedInstance() .finishBroadcast()
    }

    func broadcastFinished(_ broadcast: TXReplayKitExt, reason:
TXReplayKitExtReason) {
        var tip = ""
        switch reason {
        case TXReplayKitExtReason.requestedByMain:
            tip = "屏幕共享已结束"
            break
        case TXReplayKitExtReason.disconnected:
            tip = "应用断开"
            break
        case TXReplayKitExtReason.versionMismatch:
            tip = "集成错误（SDK 版本号不相符合）"
            break
        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:
            // 处理视频样本缓冲区
            TXReplayKitExt.sharedInstance()
.sendVideoSampleBuffer(sampleBuffer)
            break
        case RPSampleBufferType.audioApp:
            // 处理应用程序音频的音频样本缓冲区
            break
        case RPSampleBufferType.audioMic:
```

```
                // 处理麦克风音频的音频样本缓冲区
                break
            @unknown default:
                // 处理其他样本缓冲液类型
                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	配置信息。
callback	TCICCallback	否	回调函数。

TCICConfig 字段信息

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	名称配置。
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	组件配置列表。

TCICComponentConfig 字段信息

TCICComponentConfig 是一个抽象类，用于配置课堂界面中的各个组件。通过将具体的组件配置类实例添加到 componentConfig 列表中，可以实现对各个组件的自定义配置。

组件配置使用示例

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

// 配置参数
final config = TCICConfig(
  userId: 'user123',
  classId: 'class456',
  role: RoleEnum.student,
  token: 'your_token_here',
  // 组件配置列表
  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(),
  ],
);

// 根据您的业务场景，跳转到课堂页面
```

```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => TCICView(
      controller: controller,
      config: config,
    ),
  ),
);
```

HeaderComponentConfig 字段信息

HeaderComponentConfig 用于配置头部组件，其字段信息如下：

字段名	类型	是否必填	描述
isShow	bool	否	是否显示头部组件（默认为 true）。
enableHandsUp	bool	否	是否显示举手（默认为 true）。
enableScreenShare	bool	否	是否显示屏幕共享（默认为 true）。
enableMessage	bool	否	是否显示消息（默认为 true）。
enableCourseware	bool	否	是否显示课件（默认为 true）。
enableSetting	bool	否	是否显示设置（默认为 true）。
enableMemberList	bool	否	是否显示花名册（默认为 true）。
showClassStatus	bool	否	是否显示课程状态（默认为 true）。
showClassTime	bool	否	是否显示课程时间（默认为 true）。
showOnlineMemberCount	bool	否	是否显示在线成员数量（默认为 true）。
showQuitButton	bool	否	是否显示退出按钮（默认为 true）。
showClassLogo	bool	否	是否显示课程 Logo（默认为 true）。
showClassName	bool	否	是否显示课程名称（默认为 true）。
showClassInfo	bool	否	是否显示课程信息（默认为 true）。

showNetworkStatus	bool	否	是否显示网络状态（默认为 true）。
iconConfig	HeaderIconConfig	否	自定义头部组件图标配置。
headerBuilder	Widget Function()	否	自定义整个头部组件。
headerLeftBuilder	Widget Function()	否	自定义头部组件左侧组件。
headerRightBuilder	Widget Function()	否	自定义头部组件右侧组件。
headerActionsBuilder	Widget Function()	否	自定义头部组件中间组件。

HeaderIconConfig 字段信息

HeaderIconConfig 用于配置头部组件的图标，其字段信息如下：

字段名	类型	是否必填	描述
micIcon	String	否	麦克风图标。
micDisableIcon	String	否	麦克风禁用图标。
cameraIcon	String	否	摄像头图标。
cameraDisableIcon	String	否	摄像头禁用图标。
handUpIcon	String	否	举手图标。
screenShareIcon	String	否	屏幕共享图标。
messageIcon	String	否	消息图标。
coursewareIcon	String	否	课件图标。
settingIcon	String	否	设置图标。
membersIcon	String	否	成员图标。

MemebersComponentConfig 字段信息

MemebersComponentConfig 用于配置成员组件，其字段信息如下：

字段名	类型	是否必填	描述
enableStageUpDownAction	bool	否	是否启用上下台（默认为 true）。

MessageComponentConfig 字段信息

MessageComponentConfig 用于配置消息组件，其字段信息如下：

字段名	类型	是否必填	描述
messageItemBuilder	Widget Function(V2TimMessage)	否	消息 Item 构建器。
messageHeaderBuilder	Widget Function(V2TimMessage)	否	消息头构建器。
messageBubbleBuilder	Widget Function(V2TimMessage, Widget)	否	消息气泡构建器。
messageRowBuilder	Widget Function(V2TimMessage)	否	消息行构建器。

SettingComponentConfig 字段信息

SettingComponentConfig 用于配置设置组件，其字段信息如下：

字段名	类型	是否必填	描述
enableAudioSetting	bool	否	是否启用音频设置（默认为 true）。
enableVideoSetting	bool	否	是否启用视频设置（默认为 true）。
enableGeneralSetting	bool	否	是否启用通用设置（默认为 true）。
showMemberJoinExitInfo	bool	否	是否显示成员加入退出课堂信息（默认为 true）。
showMemberhandsupInfo	bool	否	是否显示成员举手信息（默认为 true）。

VideoComponentConfig 字段信息

VideoComponentConfig 用于配置视频组件，其字段信息如下：

字段名	类型	是否必填	描述

videoFloatBuilder	Widget Function()	否	视频悬浮层构建器。
videoActionBuilder	Widget Function()	否	视频组件操作按钮构建器。

WhiteboardComponentConfig 字段信息

WhiteboardComponentConfig 用于配置白板组件，其字段信息如下：

字段名	类型	是否必填	描述
enableCreateBoard	bool	否	是否启用创建白板（默认为 true）。
enableBoardList	bool	否	是否启用白板列表（默认为 true）。
enableSwitchPage	bool	否	是否 PPT 课件可翻页（默认为 true）。

TCICLangConfig 字段信息

TCICLangConfig 用于配置语言信息，其字段信息如下：

字段名	类型	是否必填	描述
lang	TranslateLangEnum	是	语言类型（zh、en、ja、ko、zhTw 等）。

TranslateLangEnum 枚举值

TranslateLangEnum 用于定义支持的语言类型，其枚举值如下：

枚举值	描述
zh	中文
en	英文
ja	日文
ko	韩文
zhTw	繁体中文
fr	法文
de	德文
es	西班牙文

ru	俄文

TCICFontConfig 字段信息

TCICFontConfig 用于配置字体信息，其字段信息如下：

字段名	类型	是否必填	描述
fontFamily	String	否	字体族名称。
fontPath	String	否	字体文件路径（相对于 assets 目录）。
fontUrl	String	否	字体文件 URL（网络字体）。
enableCustomFont	bool	是	是否启用自定义字体（默认为 false）。
fontWeights	Map<FontWeight, String>	否	字体权重配置。
fontStyles	Map<FontStyle, String>	否	字体样式配置。

TCICNameConfig 字段信息

`TCICNameConfig` 用于配置不同语言下的名称显示，其字段信息如下：

字段名	类型	是否必填	描述
zh	NameConfigInfo	否	中文名称配置。
en	NameConfigInfo	否	英文名称配置。
ja	NameConfigInfo	否	日文名称配置。
ko	NameConfigInfo	否	韩文名称配置。
zhTw	NameConfigInfo	否	繁体中文名称配置。

NameConfigInfo 字段信息

`NameConfigInfo` 用于定义具体语言下的各项名称，其字段信息如下：

字段名	类型	是否必填	描述

teacher	String	否	老师
assistant	String	否	助教
student	String	否	学生
host	String	否	主持人
coHost	String	否	副主持人
panelist	String	否	panelist
viewer	String	否	观众
raiseHand	String	否	举手
lowerHand	String	否	放下手
chat	String	否	聊天
whiteboard	String	否	白板
fileShare	String	否	文件分享
screenShare	String	否	屏幕分享

回调函数

TCICCallback 支持以下回调：

- `onJoinedClassSuccess`：加入课堂成功
- `afterExitedClass`：退出课堂后
- `onJoinedClassFailed`：加入课堂失败
- `onKickedOffClass`：被踢出课堂
- `onMemberJoinedClass`：成员加入课堂
- `onMemberLeaveClass`：成员离开课堂
- `onReceivedMessage`：收到消息
- `beforeExitedClass`：退出课堂前（返回 false 可取消退出）
- `beforeRenderMessage`：渲染消息前（返回 false 可不渲染）

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。

- 运行时权限需要在代码中动态申请。
- 某些权限（如相机、麦克风）需要用户手动授权。

iOS 权限被拒绝：

- 检查 `Info.plist` 中是否已添加权限描述。
- 确保权限描述文本清晰明了。
- 某些权限首次使用时会弹出授权对话框。

2. 音视频问题

无法听到声音：

- 检查设备音量设置。
- 确认麦克风权限已授权。
- 检查音频设备连接状态。

无法看到视频：

- 检查相机权限已授权。
- 确认设备相机功能正常。
- 检查网络连接状态。

3. 网络问题

连接失败：

- 检查网络连接状态。
- 确认防火墙设置。
- 验证 SDKAppId 和 token 是否正确。

音视频卡顿：

- 检查网络带宽是否充足。
- 确认设备性能是否满足要求。
- 考虑降低音视频质量设置。

4. 白板问题

白板无法加载：

- 检查网络连接状态。
- 确认白板服务是否正常。
- 验证白板配置参数。

白板操作无响应：

- 检查触摸事件是否正常。
- 确认白板组件状态。
- 验证用户权限设置。

技术支持

如果您在使用过程中遇到问题，可以通过以下方式获取技术支持：

- 腾讯云实时互动-教育版官方文档
- 腾讯云开发者社区
- GitHub Issues

uni-app

最近更新时间：2025-08-13 17:33:22

开发环境要求

请根据 uni-app 原生语言插件 使用教程，学习如何在您的 uni-app 项目中引入原生语言插件。

- iOS 版本要求：11+
- Android API 要求：21+

前提条件

您已 注册腾讯云 账号，并完成 实名认证 。

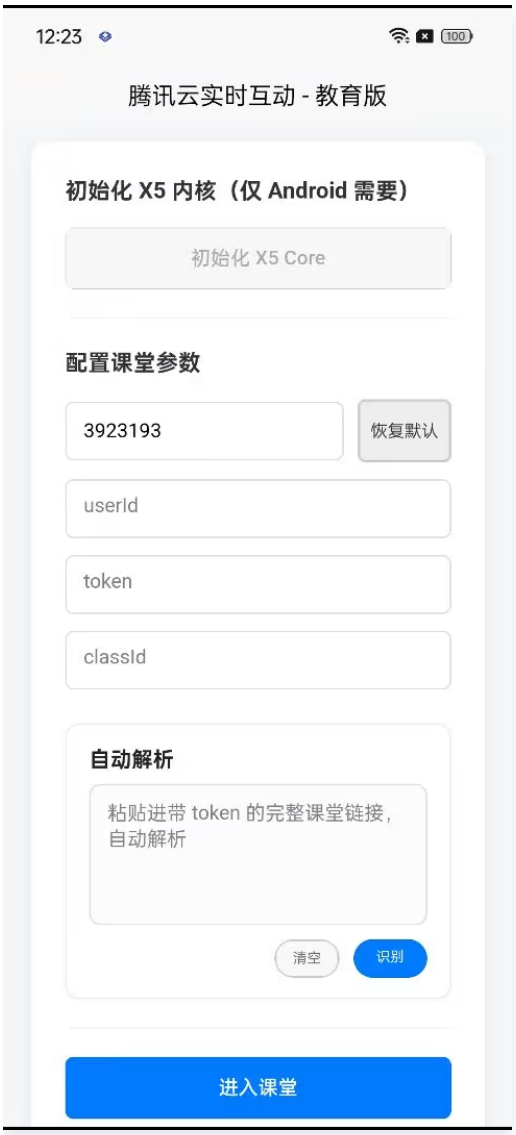
Demo 及源码

为了帮助您更好地了解一个集成了腾讯云实时互动-教育版 uni-app 插件的应用，我们准备了 Demo 及配套源码供您参考。您可以查看 GitHub 获取源码 。

直接安装体验 Demo

若您使用 Android 设备，可扫码下载安装我们打包好的 uni-app 项目直接体验进入课堂。

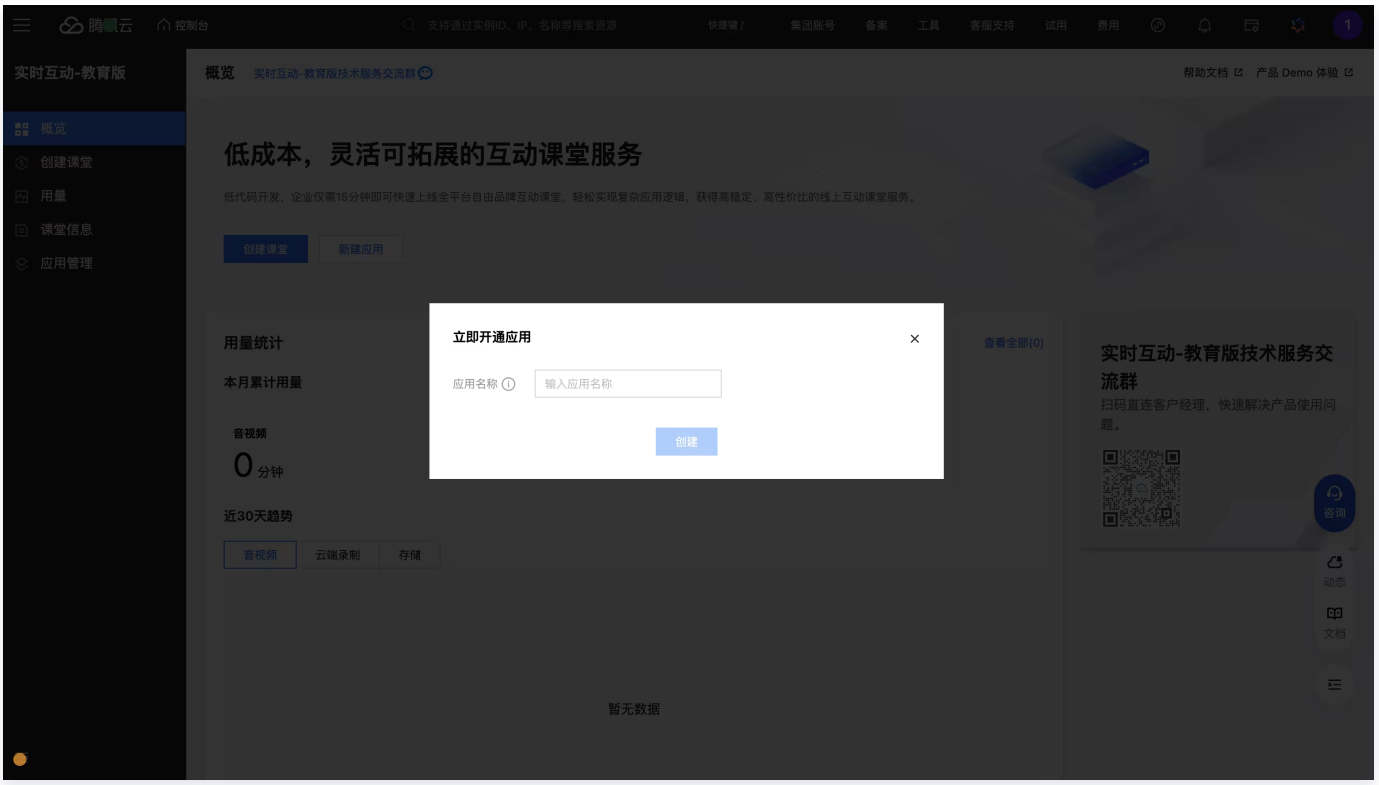
进入应用后，如下截图所示。您可直接复制我们 线上 Demo 创建的课堂 信息，进入 uni-app native 版本课堂。
也可直接将课堂链接 (需要手动在末尾添加 &token=${通过 F12 打开 Console 进入网络请求 Network 内课堂请求 payload 来查看 token}) **贴入并自动解析参数。**



操作步骤

步骤一：创建新的应用

1. 登录 实时互动-教育版 控制台，进入左侧导航栏的**概览**，选择**新建应用**。
2. 若尚未创建应用，则默认进入“创建应用”界面，输入应用名称，例如 TestLCIC。



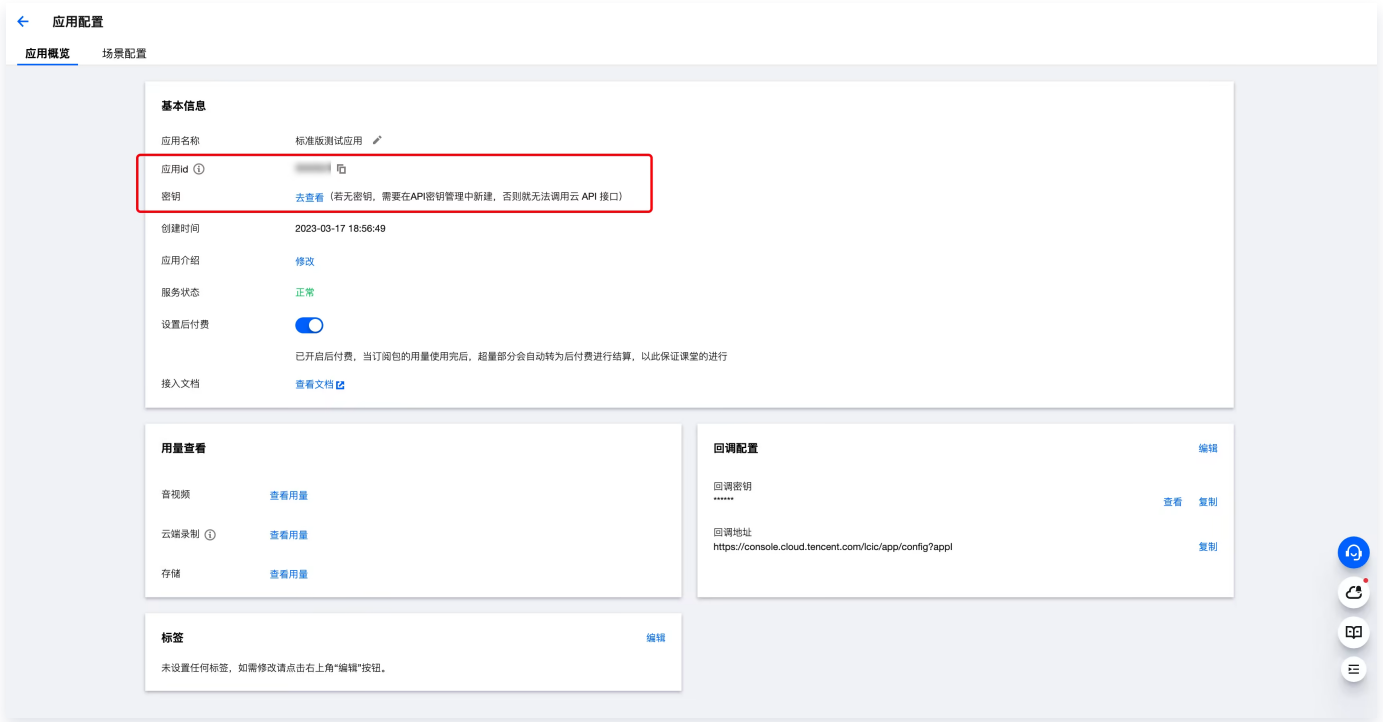
若您已创建应用，可前往**应用管理**中，选择使用已有应用即可。

说明：

- 移动端需要购买旗舰版或企业尊享版后方可接入。若需创建商用应用，可根据业务需求在 购买页 创建对应版本的应用。
- 应用名称只允许下划线、数字或中英文字符。

步骤二：获取 SDKAppId 和密钥(SecretKey)

1. 进入**应用管理 > 应用配置**，获取 SDKAppId 。
2. 进入 访问管理(CAM)控制台 获取密钥，若无密钥，需要在 API 密钥管理中进行新建，具体可参见 访问密钥管理 。



步骤三：导入 SDK

1. 前往购买 TRTC 免费插件 **腾讯云实时音视频SDK** (插件1)，并选择该插件绑定的项目。
2. 通过 GitHub 仓库 ，获取**腾讯云实时互动-教育版 uniapp 插件包** (插件2)，将其中 `lcic-sdk-pro` 本地引入，放置在项目 nativeplugins 文件夹下。如果项目没有该文件夹，请手动创建一个。
3. 在 HBuilderX 里找到项目，在 manifest 的 app 原生插件配置中勾选如上两个插件(插件1 & 插件2)，如下图所示。

> **更新说明：**
> 我们的 Native SDK 会不定期更新，请多关注 本项目 GitHub 仓库，如果有更新，建议下载后替换本地项目 nativeplugins 中的插件包。

步骤四：SDK 授权申请

需要您提交 腾讯云工单，向我们发送 SDK 权限申请。请按下表模板提供对应信息。在信息确认无误的情况下，我们将会在1个工作日完成。

> ⚠ **注意：**
> - 一个旗舰版仅支持授权一个正式包名，请确认无误后发送相关信息。
> - 包名用于 x5 内核以及快直播播放器签名授权，请提供所需授权的正式应用的 App Name、Package Name 和 Bundle ID 信息。

分类	说明
问题标题	实时互动-教育版 Android SDK 授权申请
问题主要内容	公司名称。如，xxx 有限公司。
	个人姓名
	联系方式
	App Name
	Package Name（Android）
	Bundle ID（iOS）

步骤五：初始化 X5 内核

X5 内核相对于系统 WebView，具有兼容性更好，速度更快等优势。Android 实时互动-教育版 SDK 的组件实现依赖于 X5 内核的 WebView。现提供 X5 内核静态集成方式，能提升 X5 内核加载成功率且无需进程重启即可生效。

1. 检查同意隐私政策协议。

> ⚠ **注意：**
> 建议在同意隐私政策协议之后，再调用初始化 X5 内核的方法，以免上架应用市场时出现未经用户同意收集个人信息的情况。

2. 初始化 X5 内核。进入课堂前，**必须保证该方法执行完毕**。无论成功与否，后续都可正常进入课堂。如果初始化失败，进入课堂采用兜底 WebView 方案。

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

lcicModule.initX5Core({
                        licenseKey: "" // 申请的 X5 LicenseKey
```

```
                    },
                    (ret) => {
                        if (ret && ret.code === 0) {
                            // X5 内核初始化成功. 后续进入课堂走 X5 内核方
案.
                        } else {
                            // X5 内核初始化失败. 后续进入课堂走兜底
Webview 方案.
                        }
                    }
```

> ⚠ **注意：**
> - initX5Core 中的 licenseKey 参数需要通过 步骤四 提交工单联系我们获取 X5 内核的 licenseKey。
> - 如果出现X5初始化失败，可及时 联系我们 。

步骤六：获取进入课堂所需参数

joinClass 参数解释：

- 通过 控制台 进入**应用管理 > 应用配置**，获取 SDKAppId ，即为学校编号(schoolId)信息。
- 通过云 API 接口 CreateRoom 创建课堂，可以获取到课堂号(classid)信息。
- 通过调用云 API 接口 RegisterUser 注册用户，可以获取到对应的用户 ID(userid)信息。
- 通过云 API 接口 LoginUser 登录，可以获取到用户鉴权 token 信息。
- scene、lng、camera、mic、speaker 为非必要参数，如果不设置则使用的是默认值。

字段	类型	必填	含义	备注
schoolId	int	是	学校编号	通过控制台进入**应用管理 > 应用配置**，获取 SDKAppId 。
classId	long	是	课堂编号	通过 CreateRoom 接口创建返回 RoomId 获取。
userId	string	是	用户账号	通过 RegisterUser 接口获取。
token	string	是	后台鉴权参数	通过 LoginUser 接口获取。
scene	string	否	场景名称	用于区分不同的定制布局，通过 SetAppCustomContent 接口配置。

lng	string	否	语言参数	当前支持中文(简体)、中文(繁体) 、English、韩语、日语、阿拉伯语、越南语、印尼语。可拼接相应参数，展示对应语种。参数：zh-CN、zh-TW、en-US、ka、ja、ar、vi、id。
camera	int	否	初始化开启摄像头	1为开启摄像头，0为关闭摄像头，默认 1。
mic	int	否	初始化开启麦克风	1为开启麦克风，0为关闭麦克风，默认 1。
speaker	int	否	初始化开启扬声器	1为开启扬声器，0为关闭扬声器，默认 1。

步骤七：调起组件主页面

只需传递 4 个参数就可调起 LCIC 组件主页面，分别为学校编号、课堂编号、用户账号和 token。

> **说明：**
> `schoolId` 同 SDKAppId 。

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

lcicModule.joinClass({
                        schoolId: schoolId,
                        userId: userId,
                        token: token,
                        classId: classId,
                        // 其他更多可选参数，可参考步骤六
                    },
                    (res) => {
                        // 进入课堂成功
                    }
                );
```

步骤八：运行项目

调试阶段，您可创建包含 TRTC 及腾讯云实时互动-教育版两个插件的自定义基座。
调试完成后，可使用云打包等方式，打包并导出项目。