

# 即时通信 IM

## 语音聊天室（含 UI）



腾讯云

## 【 版权声明 】

©2013–2025 腾讯云版权所有

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

## 【 商标声明 】



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

## 【 服务声明 】

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

## 【 联系我们 】

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



## 文档目录

### 语音聊天室（含 UI）

- 组件介绍（TUILiveKit）

- 开通服务（TUILiveKit）

- 跑通 Demo（TUILiveKit）

  - iOS

  - Android

- 快速接入（TUILiveKit）

  - Android

  - iOS

- 界面定制（TUILiveKit）

- 开播和收听（TUILiveKit）

  - Android

  - iOS

- 麦位管理（TUILiveKit）

- 房间列表（TUILiveKit）

- 关注主播（TUILiveKit）

- 互动弹幕（TUILiveKit）

- 互动礼物（TUILiveKit）

- 礼物特效（TUILiveKit）

- 客户端API（TUILiveKit）

  - Android

    - VoiceRoomKit

    - SeatGridView

    - SeatGridViewObserver

    - VoiceRoomDefine

  - iOS

    - VoiceRoomKit

    - SeatGridView

    - SeatGridViewObserver

    - VoiceRoomDefine

- 服务端 API

  - 账号系统

    - 账号管理

    - 登录鉴权

    - 生成 UserSig

    - 资料管理

## REST API

### REST API 简介

### REST API 列表

#### 用户管理

##### 账号管理

###### 导入单个账号

###### 导入多个账号

###### 查询账号

##### 用户资料

###### 设置资料

###### 拉取资料

#### 房间管理

##### 创建房间

##### 更新房间

##### 获取房间信息

##### 获取直播列表

##### 解散房间

##### 获取房间自定义信息

##### 设置房间自定义信息

##### 删除房间自定义信息

#### 房间成员管理

##### 成员禁言

##### 获取禁言列表

##### 成员封禁

##### 解除成员封禁

##### 修改管理员

##### 获取封禁列表

##### 获取管理员列表

##### 获取房间成员列表

#### 连麦管理

##### 获取麦位列表

##### 用户上麦

##### 用户下麦

##### 锁定麦位

#### PK 相关

##### 发起 PK

##### 修改 PK 分数

##### 获取 PK 信息

#### 云端搜索

房间成员搜索

第三方回调

第三方回调简介

回调命令列表

回调配置

查询回调配置

设置回调配置

更新回调配置

删除回调配置

房间相关

创建房间之后回调

解散房间之后回调

更新房间信息之后回调

设置房间自定义信息后回调

删除房间自定义信息后回调

用户相关

房间成员状态变更回调

连麦相关

麦位列表变更之后回调

PK 相关

创建 PK 之后回调

PK 开启之后回调

PK 结束之后回调

错误码（TUILiveKit）

发布日志（TUILiveKit）

iOS

Android

常见问题（TUILiveKit）

iOS

Android

# 语音聊天室（含 UI）

## 组件介绍（TUILiveKit）

最近更新时间：2025-07-14 15:13:52

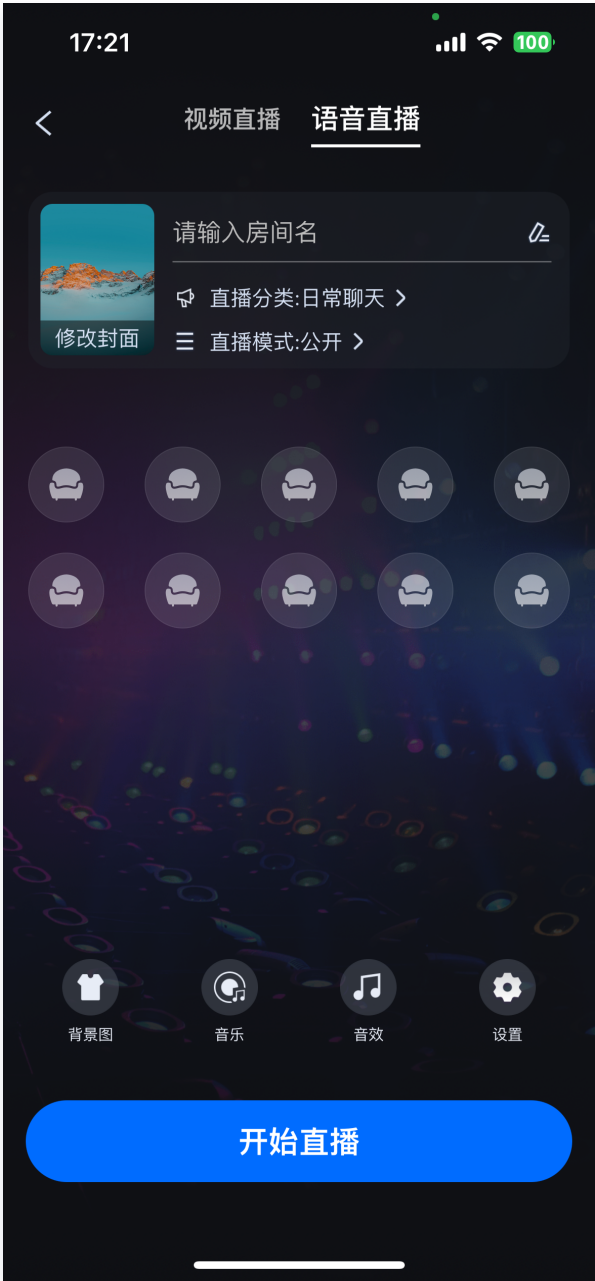
### 组件介绍

TUILiveKit 是一款适用于社交娱乐、购物、健身等互动直播场景的产品，通过集成该产品，仅需三步，30分钟内就可以为您的 App 添加互动连麦、送礼、房间管理等功能，快速上线业务。基本功能展示如下图：



### 主播端

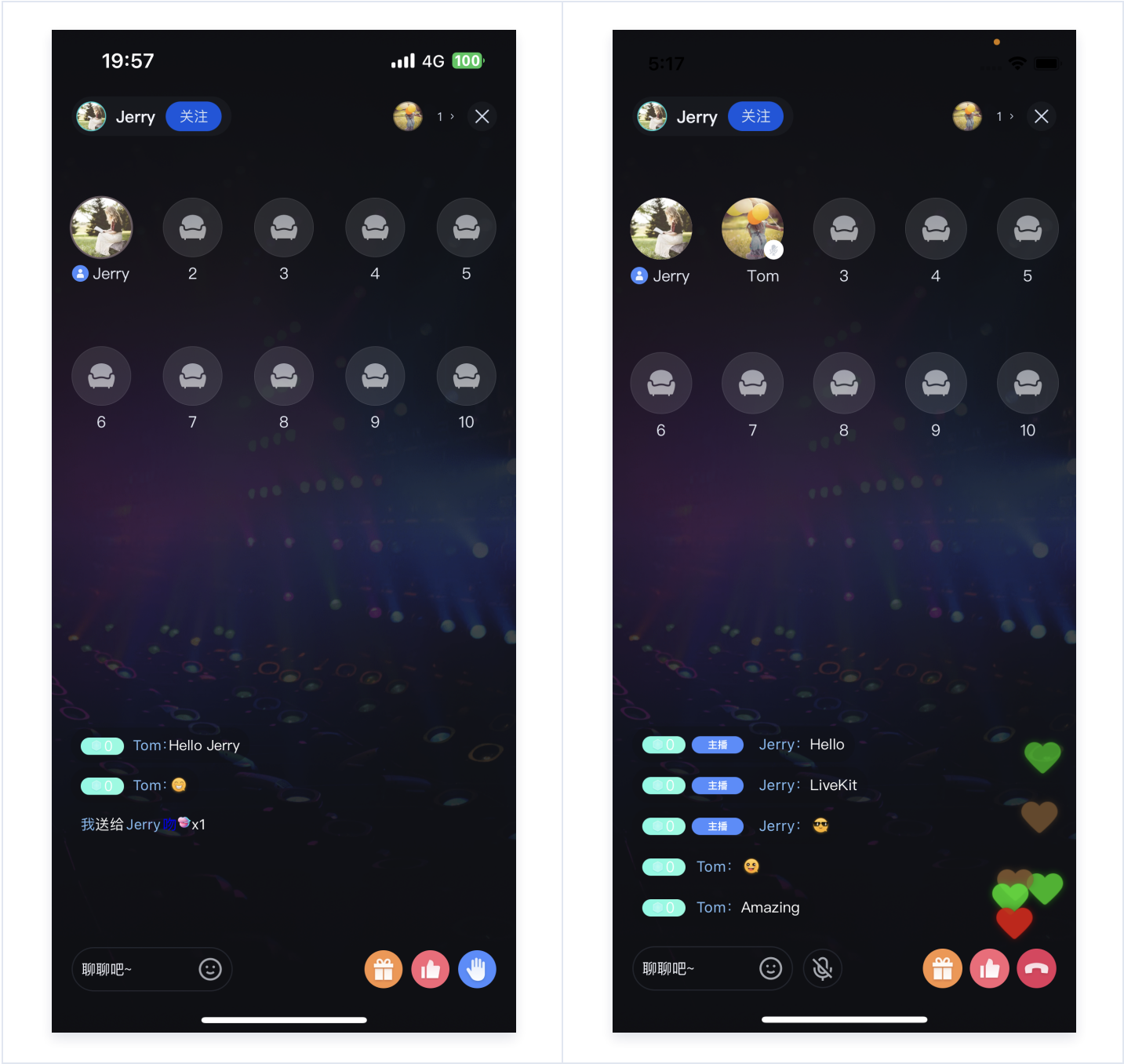
|         |        |
|---------|--------|
| 语音直播开播页 | 语音直播主页 |
|---------|--------|



观众端

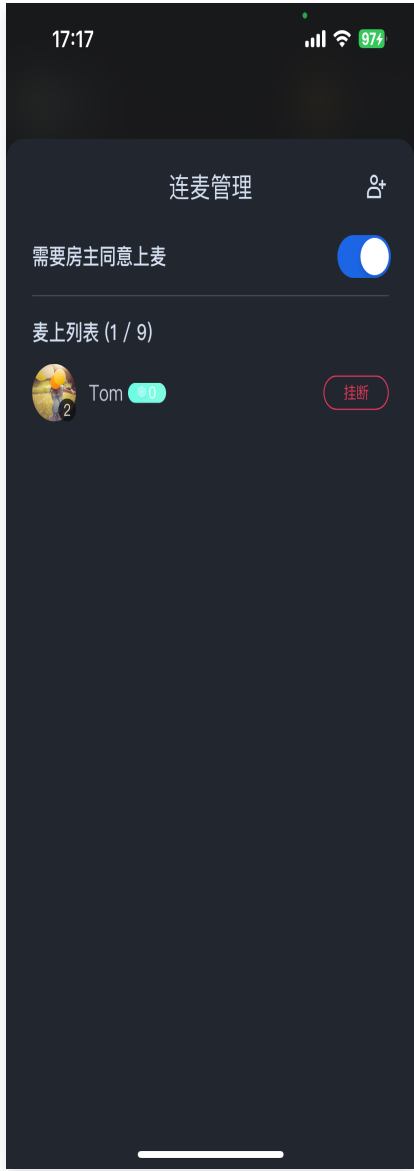
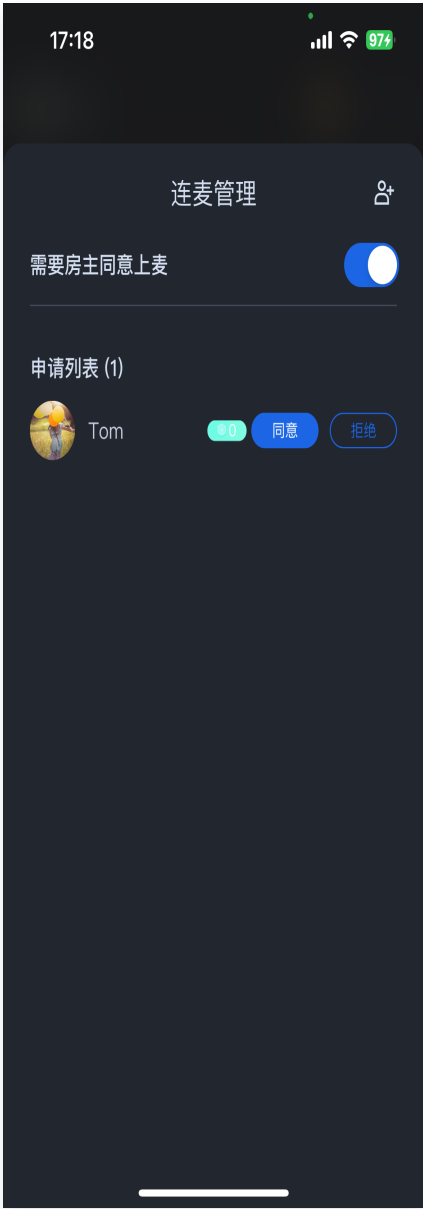
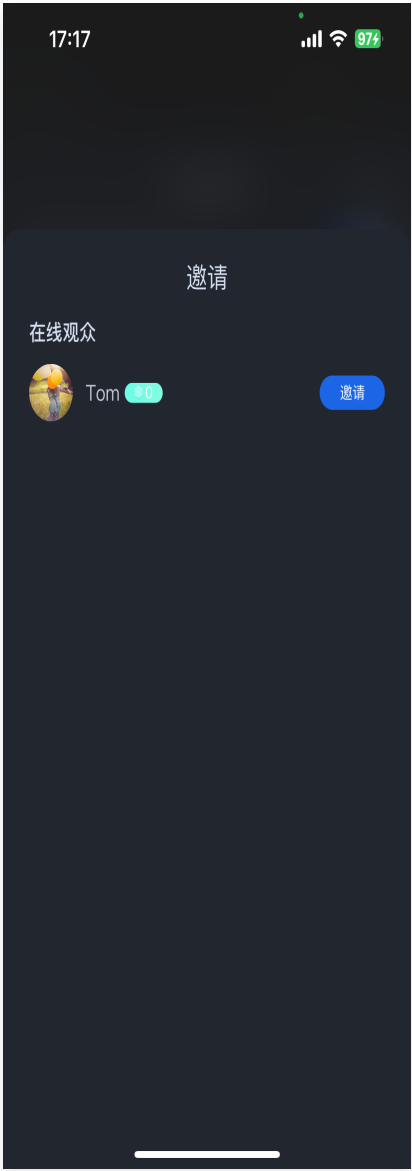
语音直播未上麦

语音直播上麦后



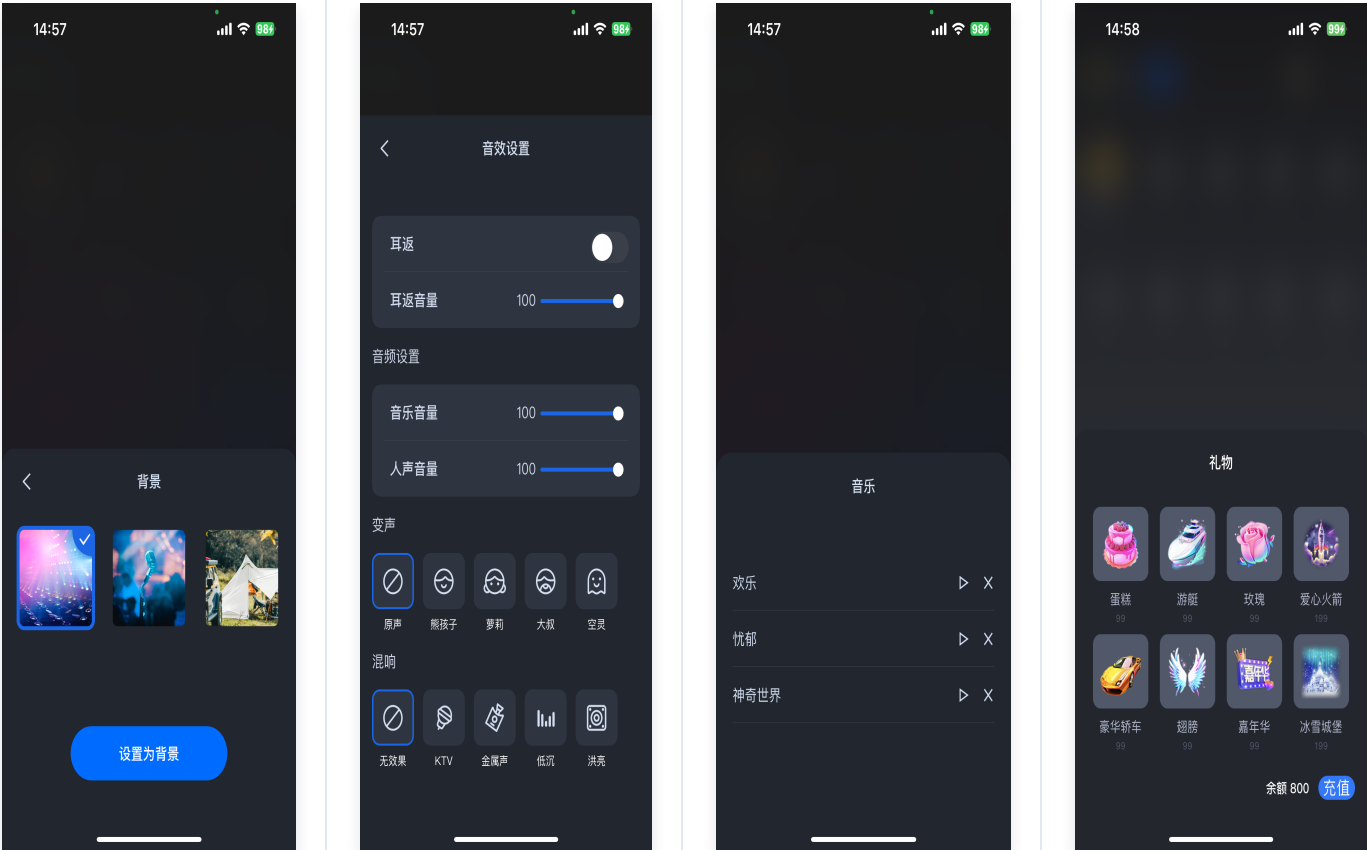
麦位管理

|        |        |      |
|--------|--------|------|
| 邀请上麦管理 | 申请上麦管理 | 麦位管理 |
|--------|--------|------|



高级功能

|       |       |    |            |
|-------|-------|----|------------|
| 直播间背景 | 音效&变声 | 音乐 | 互动礼物（全屏礼物） |
|-------|-------|----|------------|



支持平台

| 平台      | Android                                                                 | iOS                                                                           |
|---------|-------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| 是否支持    | ✓                                                                       | ✓                                                                             |
| 支持语言/框架 | <ul style="list-style-type: none"><li>• Java</li><li>• Kotlin</li></ul> | <ul style="list-style-type: none"><li>• Swift</li><li>• Objective-C</li></ul> |

功能介绍

基础功能

| 功能    | 功能描述                             |
|-------|----------------------------------|
| 语音聊天  | 主播可以在开启语音聊天室前设置背景和音效，支持开启和关闭聊天室。 |
| 观看语聊房 | 支持多人同时观看语聊房。                     |
| 单房间连麦 | 支持单房间内与多位观众连线，以及跨房与其他多位主播连线。     |



|         |                                                |
|---------|------------------------------------------------|
| 跨房连线/PK | 支持双人、多人跨房 PK。                                  |
| 在线观众列表  | 观众列表组件支持展示语聊房最新进房的前100位观众及总人数，提供简化版和详情版视图展示功能。 |
| 聊天弹幕    | 用户可发送表情和文本消息弹幕，实时展示于语聊房内。                      |
| 心动点赞    | 支持点赞功能，并统计基础点赞数据。                              |
| 礼物特效    | 提供弹幕动画、全屏动画（支持 SVGA 格式）及点赞动画的播放功能。             |

高级功能

| 功能             | 功能描述                                              |
|----------------|---------------------------------------------------|
| 成员管理：拉黑禁言      | 支持主播对观众进行拉黑和禁言操作。                                 |
| 礼物系统           | 展示礼物面板，支持配置礼物名称、价格、等级等信息。                         |
| 关注             | 观众可以在直播间关注主播及其他观众。                                |
| 直播间搜索          | 支持搜索直播间内的成员。                                      |
| 臻臻画质           | 在保证画质的前提下显著降低码率，或在相同比特率下提升画质，优化带宽受限环境下的观看体验。      |
| PC推流助手         | 支持多机位画面采集和本地音频混流，适用于 PC 端专业推流。                    |
| RTMP推流/输入在线媒体流 | 通过 RTMP 协议将本地音频推送至语音聊天室，支持主播 OBS 开播、虚拟人直播等多元场景。   |
| QUIC弱网稳定传输     | 可以根据终端的网络状况，自动选择切换网络链路或多链路传输，从而在弱网环境下提升传输速度。      |
| 腾讯特效           | 在默认美颜基础上，支持集成 <a href="#">腾讯特效</a> ，提供更全面高级的美颜功能。 |
| 音效变声           | 支持开播前设置音量、变声和混响效果。                                |

- ❗ 说明：
- 单房间连麦时预制视频直播 UI 为1-9人连麦效果，语聊 UI 为10人效果；超出对应人数的连麦 UI 可自行绘制。
  - 更多功能与计费说明参见 [开通服务](#)。

在线体验

|    |         |     |
|----|---------|-----|
| 平台 | Android | iOS |
|----|---------|-----|

[demo 集成](#)[Github: Andorid](#)[Github: iOS](#)

# 开通服务（TUILiveKit）

最近更新时间：2025-07-03 12:02:54

本文介绍如何 [免费开通体验版](#)、[开通正式版](#) 和 [续费正式版](#) TUILiveKit 服务，您可以根据 [TUILiveKit 能力版本计费说明](#) 选择您需要版本，并参见下述指引免费开通和购买开通正式版 TUILiveKit。

## 开通体验版

TUILiveKit 是基于腾讯云 [即时通信 IM](#) 和 [实时音视频 TRTC](#) 两项付费 PaaS 服务构建出的语音聊天室组件。为了您能更好地体验语音聊天室功能，我们免费为每个 SDKAppID 提供了语音聊天室能力7天体验版（体验版不额外赠送通话时长），每个 SDKAppID 可免费体验2次，每次有效期均为7天；同时一个账号下所有 SDKAppID 的体验总次数为10次。

1. 登录 [即时通信 IM 控制台](#)，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。

### ❗ 说明：

Live 目前仅支持选择中国、新加坡和硅谷数据中心，如需使用其他部署区域请 [联系我们](#)。

创建新应用

应用名称 \*

Test

数据中心 ⓘ \*

中国 数据存储在中国，支持全球接入

标签 ⓘ

标签键

标签值

+ 添加

📎 键值粘贴板

确定

2. 创建后即可在应用列表查看所有应用，这里的 `SDKAppID`、`密钥` 会在后续步骤中使用到。

应用管理 IM 技术服务交流群 IM 出海交流专区

产品体验，你说了算

创建新应用

请输入 SDKAppID 或 应用名称 或 标签

| 应用名称 | SDKAppID | 应用状态 | 数据中心 | 标签 | 创建时间       | 服务信息                                                                                    | 服务状态               | 到期时间                 | 操作                                        |
|------|----------|------|------|----|------------|-----------------------------------------------------------------------------------------|--------------------|----------------------|-------------------------------------------|
|      |          | 正常   | 中国   | -  | 2025-03-21 | <a href="#">消息服务 Chat 体验版</a><br><a href="#">推送服务 Push</a><br><a href="#">智能客服 Desk</a> | 使用中<br>未开通<br>未开通  | -<br>-<br>-          | <a href="#">应用详情</a> <a href="#">查看密钥</a> |
|      |          | 正常   | 新加坡  | -  | 2025-03-12 | <a href="#">消息服务 Chat 体验版</a><br><a href="#">推送服务 Push</a><br><a href="#">智能客服 Desk</a> | 使用中<br>未开通<br>未开通  | -<br>-<br>-          | <a href="#">应用详情</a> <a href="#">查看密钥</a> |
|      |          | 正常   | 中国   | -  | 2025-03-12 | <a href="#">消息服务 Chat 旗舰版</a><br><a href="#">推送服务 Push</a><br><a href="#">智能客服 Desk</a> | 自动续费<br>未开通<br>未开通 | 2025-04-21<br>-<br>- | <a href="#">应用详情</a> <a href="#">查看密钥</a> |

3. 在应用列表单击刚创建的应用，进入应用详情页面，再点击左侧导航栏音视频服务 RTC，进入刚创建应用的音视频服务管理页面。

即时通信 IM

应用管理

功能服务

消息服务 Chat

推送服务 Push

客服服务 Desk

音视频服务 RTC

通用工具

回调配置

UserSig生成校验

自助排障日志

应用详情

当前数据中心: 中国

IM 技术服务交流群

IM 出海交流专区

产品体验，你说了算

当前应用未开启内容审核，建议开通，帮助业务降低内容风险。立即开通

应用信息

修改应用 更多操作

应用名称: Test

SDKAppID:

状态: 正常

应用类型: 其它

创建时间: 2025-03-21

标签:

应用简介: -

服务管理

消息服务 Chat

管理

版本: 体验版

服务状态: 使用中

到期时间: -

升级套餐包

申请开发版

推送服务 Push

未开通

3 分钟集成，精准推送，毫秒级触达，多通道多平台覆盖

全生命周期数据统计与排查，轻松提升用户留存 10% - 60%

立即购买

免费试用

智能客服 Desk

未开通

提供智能机器人与客服工作台，提高客户服务效率

自定义问答策略及客服接待流程，提升客户服务体验

立即购买

免费试用

音视频服务 RTC

未开通

RTC 可以帮助您在 IM 应用中实现音视频通话、多人语聊、视频会议等功能。RTC 独立计费，详见 [价格总览](#)。

立即前往

体验 Demo

4. 勾选协议，点击立即开通，即可开通实时音视频 TRTC 服务。



5. 在**含 UI 低代码场景方案 > 直播/语聊房 (TUILiveKit)**，单击**免费体验**即可开通 TUILiveKit 的 7 天免费试用服务。

**说明：**

- 领取体验版后，将解锁全部 TRTC 功能。
- 如果需要**正式应用上线**，可以单击 [购买正式版](#) 即可进入购买页面。
- IM 音视频能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 [IM 购买页](#) 了解包含功能并选购您适合的版本。

音视频服务 RTC

当前数据中心: 中国 ①

[IM 技术服务交流群](#)[IM 出海交流专区](#)[产品体验, 你说了算](#)

## 音视频服务

## 1 选择音视频场景

音视频服务提供音视频通话、多人会议、在线直播/语音聊天等含 UI 组件快速接入方式, 如需自定义 UI 可尝试无 UI 接入方式, [查看详情](#)

## 2 快速接入音视频通话能力

10 分钟快速跑通代码, 几行代码将方案正式集成到您的项目中

## 3 购买正式版资源

我们提供多个版本的包月套餐, 满足您的业务需要。详情查看[计费说明](#)

## 4 质量监控&amp;运营统计

集成音视频服务后, 可第一时间查询通话质量详情以及查看[用量统计](#)、[监控仪表盘](#)

## 含 UI 低代码场景方案

音视频通话 (TUICallKit) 多人音视频 (TUIRoomKit) **在线直播/语音聊天室(TUILiveKit)**

TUILiveKit 是 IM 与 TRTC 联合打造的直播组件, 适用于语音/视频直播或语聊房场景 (参考[微信视频号](#)), 查看[场景方案](#)介绍。

## 基本信息

服务状态 **未开通**

当前版本 -

[购买正式版](#)[免费体验](#)

## 快速接入

10 分钟快速跑通代码, 几行代码将方案正式集成到您的项目中

[Android 集成文档](#)[iOS 集成文档](#)[Flutter 集成文档](#)[Electron 集成文档](#)[Android 集成文档](#)[iOS 集成文档](#)

## Demo 体验

在线直播、语音聊天室

 Android Demo iOS Demo

6. 在弹窗中单击**免费试用7天**, 即可成功开通语音聊天室免费体验版。领取完成后, 可以看到体验版的基本信息, 包括服务状态、版本信息和功能详情、到期时间。开通完成后即可参见 [集成指引](#) 进行集成。

## 免费领取含 UI 低代码场景方案体验版

X

当前应用可免费领取 7 天的含 UI 低代码场景方案体验版，可解锁所有场景方案（音视频通话 TUICallKit、多人音视频 TUIRoomKit、在线直播/语音聊天室）和 TRTC 旗舰版功能。

体验版详情 有效期 7 天

## 音视频通话（TUICallKit）（实现1对1语音和视频通话、群组通话功能）

[查看 UI 示例](#)

- |           |           |                 |             |
|-----------|-----------|-----------------|-------------|
| ✓ 语音、视频通话 | ✓ 微信同款UI  | ✓ 通话呼叫/接听/拒绝/挂断 | ✓ 通话悬浮窗     |
| ✓ 通话呼叫通知  | ✓ 自定义呼叫铃声 | ✓ 通话状态展示        | ✓ 同平台多端登录通话 |

## 多人音视频（TUIRoomKit）（实现多人视频会议、多人语音等功能）

[查看 UI 示例](#)

- |          |        |        |               |
|----------|--------|--------|---------------|
| ✓ 标准会议UI | ✓ 房间管理 | ✓ 屏幕分享 | ✓ 高分辨率画质2K/4K |
| ✓ 会议悬浮窗  | ✓ 麦位管理 | ✓ 成员管理 | ✓ 同平台多端登录会议   |

## 在线直播/语音聊天室（TUILiveKit）（实现语音视频直播、送礼弹幕等功能）

[查看 UI 示例](#)

- |        |        |        |          |
|--------|--------|--------|----------|
| ✓ 视频直播 | ✓ 语聊房  | ✓ 聊天弹幕 | ✓ 点赞送礼   |
| ✓ 多人连麦 | ✓ 连麦管理 | ✓ 观众管理 | ✓ PC推流助手 |

每个应用可免费领取 2 次体验版试用，同一账号下最多可以领取 10 次体验版试用。  
使用音视频通话、云端录制与混流等功能产生的用量会产生相应费用，详见 [计费说明](#)

免费试用 7 天

## ⚠ 注意：

- 领取体验版后仅开通 TUILiveKit 7天的体验资格，测试过程中所产生的音视频时长等资源消耗，仍会按照实时音视频 TRTC 标准计费规则计费。
- 每个 SDKAppID 可免费体验2次，每次有效期均为7天；同时一个账号下所有 SDKAppID 的体验总次数为10次。
- 新账号首次可前往 [试用中心](#) 免费领取10000分钟音视频时长。
- 如果所选 SDKAppID 体验版领取次数已达上限，需要购买 TUILiveKit 包月套餐才能开通服务，请点击[场景套餐订阅按钮](#)或前往 [购买页](#) 购买。

7. 若您当前 SDKAppID 领取的体验版已到期，您可单击[版本详情](#)，在版本详情弹窗中单击[体验续期](#)直接使用第二次体验机会，此时体验版有效期将延长7天。

## 在线直播/语音聊天室(TUILiveKit)

✕

当前版本 体验版

到期时间 2025-03-28 (剩余 7 天) **体验续期**

| 项目                  |             |
|---------------------|-------------|
| 最大观众数: 100 ⓘ        | 连麦人数: 20人 ⓘ |
| 直播间数量: 10           | 基础送礼 ⓘ      |
| 基础美颜 ⓘ              | 背景音乐        |
| 观众列表                | 观众标签        |
| PC推流助手 (Electron) ⓘ | 观众连麦        |
| 高级送礼 ⓘ              | AI 降噪 ⓘ     |
| 高分辨率画质2K/4K         |             |

[集成指引](#)**⚠ 注意:**

通过**体验续期**延长体验版有效期，将消耗1次免费体验机会。

## 开通正式版

您可参见 [TUILiveKit 能力版本计费说明](#) 确认所需要使用的版本，并参见下述内容购买和开通 TUILiveKit 正式版。

1. 登录 [即时通信 IM 购买页](#)。
2. 根据实际业务需求选择数据中心，以及需要购买语音聊天室能力的 SDKAppID，购买后无法修改。
3. 若您当前 IM 应用已为专业版、旗舰版或企业版，可跳过该步骤；否则您需要先选择您所需要的套餐包，IM 基础套餐包区别可参见 [基础服务计费说明](#)。

**❗ 说明:**

Live 目前仅支持选择中国、新加坡和硅谷数据中心，如需使用其他部署区域请 [联系我们](#)。



数据中心

中国

新加坡

法兰克福

首尔

硅谷

雅加达

数据存储在所选的数据中心，应用创建后，数据中心不可修改，请谨慎选择。  
建议业务访问者大多来自中国大陆以外地区的客户选择境外数据中心，提供覆盖更广的全球加速节点。[如何选择数据中心？](#)

SDKAppID 

创建新应用

注意: 有多个SDKAppID时，请在购买前 确认 SDKAppID 是否选择正确，购买后无法修改。

选购配置

1 选择套餐包

2 加购内容审核服务

3 加购插件服务

选择套餐包

|      | 对比项                  | 企业版 <div>推荐</div> | 旗舰版  | 专业版 |
|------|----------------------|-------------------|------|-----|
| 高级功能 | 同平台多设备在线             | 10个               | 3个   | /   |
|      | 本地消息搜索 (Android&iOS) | ✓                 | ✓    | /   |
|      | 直播群成员查看入群前消息         | ✓                 | ✓    | /   |
|      | 群消息已读回执              | ✓                 | ✓    | /   |
|      | 群定向消息                | ✓                 | ✓    | /   |
|      | 社群超大房间               | 100万人             | 10万人 | /   |
|      | 会话标记 (UI)            | ✓                 | 可加购  | 可加购 |
|      | 会话分组 (UI)            | ✓                 | 可加购  | 可加购 |
|      | 语音转文字                | ✓                 | 可加购  | 可加购 |
|      | 云端搜索                 | ✓                 | 可加购  | 可加购 |
|      | 文本消息翻译               | ✓                 | 可加购  | 可加购 |
|      | <div>展开</div>        |                   |      |     |

当前 SDKAppID 的套餐包为 **体验版**，可购买旗舰版、专业版、企业版，不可单独购买功能包。购买旗舰版、专业版、企业版后，超出套餐包免费额度的 DAU 和峰值群组数将产生 **超量费用**

4. 在可选增值功能中勾选**音视频能力**，在展开页面的加购类型选择**在线直播/语音聊天室（TUILiveKit）**，并选择您需要开通的套餐版本。语音聊天室套餐包能力详情可参见 [TUILiveKit 能力版本计费说明](#)。

可选增值功能

单个用户可加入群组数上限

1000个群/人 (免费)

3000个群/人

单个群成员数上限 (非直播群)

2000人/群 (免费)

6000人/群

延长历史消息存储时长 30 天免费，不支持预付费用购买。请参考 [功能配置](#) 文档自助修改配置，产生费用按后付费收取。

音视频能力

☒ 加购微信同款音视频通话能力 或 在线会议多人音视频互动能力 (提供标准UI，助您快速集成)

加购“音视频能力”

选择类型“在线直播/语音聊天室 (TUILiveKit)”

加购类型

音视频通话 (TUICallKit)  
适用于音视频通话、聊天应用、视频客服、医疗问诊等轻量级音视频通话场景。查看 UI 示例

多人音视频 (TUIRoomKit)  
适用于企业会议、在线教育、网络研讨会、远程招聘等多人音视频互动场景。查看 UI 示例

在线直播/语音聊天室 (TUILiveKit)  
适用于电商直播、语音聊天室、秀场直播、直播课等直播场景。查看 UI 示例

加购版本

| 对比项                  | 大规模直播版<br>直播间人数无限制，适用于大规模直播场景 | 多人连麦版<br>适用于有连麦需求或语聊房的场景 | 单主播版<br>适用于无需连麦的小规模直播场景 |
|----------------------|-------------------------------|--------------------------|-------------------------|
| 赠送音视频时长              | 140万分钟/月                      | 38万分钟/月                  | 11万分钟/月                 |
| 直播间数量                | 500                           | 50                       | 30                      |
| 最大观众数                | 无限制                           | 2000                     | 1000                    |
| 连麦人数                 | 20人                           | 10人                      | 1人 (仅主播开播，无法连麦)         |
| 主播连线PK               | 支持9个主播同场PK                    | 支持2个主播同场PK               | /                       |
| 观众连麦                 | ✓                             | ✓                        | /                       |
| 基础送礼                 | ✓                             | ✓                        | ✓                       |
| 高级送礼                 | ✓                             | ✓                        | /                       |
| 基础美颜                 | ✓                             | ✓                        | ✓                       |
| PC推送助手 (Electron)... | ✓                             | ✓                        | ✓                       |

选择版本

赠送语音时长有效期为 1 个月。

5. 购买完成后，您可回到 [即时通信 IM 控制台](#) > [音视频服务 RTC](#)，在[含 UI 低代码场景方案](#) > [直播/语聊房 \(TUILiveKit\)](#) 功能区查看当前版本信息。开通完成后即可参见 [集成指引](#) 进行集成。

含 UI 低代码场景方案

音视频通话 (TUICallKit)

多人音视频 (TUIRoomKit)

在线直播/语音聊天室 (TUILiveKit)

TUILiveKit 是 IM 与 TRTC 联合打造的直播组件，适用于语音/视频直播或语聊房场景（参考微信视频号），查看 [场景方案](#) 介绍。

基本信息

服务状态 已开通

当前版本 大规模直播版 [版本详情](#)

到期时间 2025-04-20 [续期指引](#)

自动续费 未开启 [开启自动续费](#)

快速接入

10 分钟快速跑通代码，几行代码将方案正式集成到您的项目中

[Android 集成文档](#)

[iOS 集成文档](#)

[Flutter 集成文档](#)

[Electron 集成文档](#)

[Android 集成文档](#)

[iOS 集成文档](#)

Demo 体验

在线直播、语音聊天室

Android Demo

iOS Demo

注意：

- 购买 TUILiveKit 相当于订阅对应版本的 TRTC 包月套餐，若您已经订阅过付费版的 TRTC 包月套餐，仅需购买 IM 付费版后即可。
- 若您在满足前述付费条件后，TUILiveKit 版本状态显示待激活，可参见 [激活组件](#) 进行激活。
- 一个 SDKAppID 只可激活一个低代码组件，若您之前在使用其他组件（TUICallKit 或 TUIRoomKit），可参见 [组件切换](#) 变更为TUILiveKit；若您想同时使用多款组件，可参见 [如何同时使用多款组件](#)。

功能与计费说明

TUILiveKit 由实时音视频 TRTC 和即时通信 IM 提供底层技术支持，您需要购买指定套餐后方可使用 TUILiveKit，其中免费体验版的开通方式可参见 [开通 TUILiveKit 体验版](#)。

语音聊天室能力为 IM 增值能力之一，仅当 IM 应用为专业版、旗舰版或企业版时才可购买正式语音聊天室能力。

| 不含录制/转码/ AI 套餐               |                       |              |                                |                                 |                                 |
|------------------------------|-----------------------|--------------|--------------------------------|---------------------------------|---------------------------------|
| 对比项                          |                       | 体验版          | 单主播版                           | 多人连麦版                           | 大规模直播版                          |
| 价格                           |                       | 0元/7天        | 625元/月<br><a href="#">立即选购</a> | 1875元/月<br><a href="#">立即选购</a> | 6250元/月<br><a href="#">立即选购</a> |
| <a href="#">开通条件</a>         |                       | IM 为任意版本均可开通 | IM 为专业版、旗舰版或企业版时可开通            |                                 |                                 |
| <a href="#">对应 TRTC 包月套餐</a> |                       | TRTC 基础版     | TRTC 基础版                       | TRTC 尊享版                        | TRTC 旗舰版                        |
| 赠送资源<br>(超出后付费)              | <a href="#">音视频时长</a> | —            | 110,000分钟/月                    | 380,000分钟/月                     | 1,400,000分钟/月                   |
|                              | 日活跃用户数 (DAU)          | 根据 IM 版本而定   | 10,000 DAU                     | 10,000 DAU                      | 10,000 DAU                      |
|                              | 峰值群组数                 | 根据 IM 版本而定   | 100,000个/月                     | 100,000个/月                      | 100,000个/月                      |
| TUILiveKit 专属能力              | 直播间数量 (解散后不计数)        | 10           | 30                             | 50                              | 500                             |
|                              | 最大观众数 (单直播间)          | 100          | 1000                           | 2000                            | 无限制                             |

|                   |            |                        |             |             |
|-------------------|------------|------------------------|-------------|-------------|
| 单房间连麦人数（含主播）      | 16人        | 1人（仅主播）                | 10人         | 16人         |
| 跨房连线/PK           | 支持9个房间同场PK | /                      | 支持2个房间同场 PK | 支持9个房间同场 PK |
| 直播预览              | ✓          | /                      | /           | ✓           |
| 直播瀑布流             | ✓          | /                      | /           | ✓           |
| 关注                | ✓          | /                      | /           | ✓           |
| 直播间搜索             | ✓          | /                      | /           | ✓           |
| 礼物系统              | ✓          | /                      | /           | ✓           |
| 礼物特效              | ✓          | ✓                      | ✓           | ✓           |
| 基础美颜              | ✓          | ✓                      | ✓           | ✓           |
| 观众列表              | ✓          | ✓                      | ✓           | ✓           |
| PC 推流助手（Electron） | ✓          | ✓（限时体验）                | ✓（限时体验）     | ✓           |
| RTMP 推流/输入在线媒体流   | ✓          | /                      | ✓           | ✓           |
| QUIC 弱网稳定传输       | ✓          | /                      | /           | ✓           |
| 房间内连麦             | ✓          | /                      | ✓           | ✓           |
| 高级送礼              | ✓          | /                      | ✓           | ✓           |
| AI 降噪             | ✓          | /                      | ✓           | ✓           |
| 高分辨率画质 2K/4K      | ✓          | /                      | /           | ✓           |
| 臻画质（内测）           | /          | /                      | /           | ✓（申请内测）     |
| 即时通信 IM 功能        |            | 根据 IM 版本而定<br>详见 IM 功能 | 详见 IM 专业版功能 | 详见 IM 旗舰版功能 |

|               |                                                                                   |                      |                      |                      |
|---------------|-----------------------------------------------------------------------------------|----------------------|----------------------|----------------------|
|               |                                                                                   |                      |                      |                      |
| 实时音视频 TRTC 功能 | 详见<br>TRTC 体验<br>版功能                                                              | 详见<br>TRTC 基础<br>版功能 | 详见<br>TRTC 尊享版<br>功能 | 详见<br>TRTC 旗舰版<br>功能 |
| 超出赠送资源部分后付费   | 音视频时长后付费价格详见 <a href="#">音视频时长计费说明</a> ，DAU 与峰值群组数后付费价格详见 <a href="#">IM 计费说明</a> |                      |                      |                      |
| 支持平台          | iOS、Android、Web、Flutter、Electron（体验版和大规模直播版支持，单主播版和多人连麦版限时提供）                     |                      |                      |                      |

含录制/转码/ AI 套餐（New）

| 对比项                          |                       | 多人连麦版                           | 大规模直播版                          |
|------------------------------|-----------------------|---------------------------------|---------------------------------|
| 价格                           |                       | 2875元/月<br><a href="#">立即选购</a> | 8000元/月<br><a href="#">立即选购</a> |
| <a href="#">开通条件</a>         |                       | IM 为专业版、旗舰版或企业版时可开通             |                                 |
| <a href="#">对应 TRTC 包月套餐</a> |                       | 尊享版 Plus                        | 旗舰版 Plus                        |
| 赠送资源<br>(超出后付费)              | <a href="#">音视频时长</a> | 380,000分钟/月                     | 1,400,000分钟/月                   |
|                              | 日活跃用户数（DAU）           | 10,000 DAU                      | 10,000 DAU                      |
|                              | 峰值群组数                 | 100,000个/月                      | 100,000个/月                      |
|                              | 录制转码时长（new）           | 260,000分钟/月                     | 600,000分钟/月                     |
|                              | AI 智能识别时长（new）        | 10,000分钟/月                      | 20,000分钟/月                      |
| TUILiveKit 专属能力              | 直播间数量（解散后不计数）         | 50                              | 500                             |

|                   |              |                |
|-------------------|--------------|----------------|
| 最大观众数（单直播间）       | 2000         | 无限制            |
| 单房间连麦人数（含主播）      | 10人          | 20人            |
| 跨房连线 / PK         | 支持10个房间同场 PK | 支持20个房间同场 PK   |
| 直播预览              | /            | ✓              |
| 直播瀑布流             | /            | ✓              |
| 关注                | /            | ✓              |
| 直播间搜索             | /            | ✓              |
| 礼物系统              | /            | ✓              |
| 礼物特效              | ✓            | ✓              |
| 基础美颜              | ✓            | ✓              |
| 观众列表              | ✓            | ✓              |
| PC 推流助手（Electron） | ✓（限时体验）      | ✓              |
| RTMP推流/输入在线媒体流    | ✓            | ✓              |
| QUIC弱网稳定传输        | /            | ✓              |
| 房间内连麦             | ✓            | ✓              |
| AI 降噪             | ✓            | ✓              |
| 高分辨率画质 2K/4K      | /            | ✓              |
| 臻画质（内测）           | /            | ✓（申请内测）        |
| 即时通信 IM 功能        |              | 详见<br>IM 专业版功能 |

|               |                                                                                                                                                                                                                                                                              |                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
|               |                                                                                                                                                                                                                                                                              |                        |
| 实时音视频 TRTC 功能 | 详见<br>TRTC 尊享版 Plus 功能                                                                                                                                                                                                                                                       | 详见<br>TRTC 旗舰版 Plus 功能 |
| 超出赠送资源部分后付费   | <ul style="list-style-type: none"><li>音视频时长后付费价格详见 <a href="#">音视频时长计费说明</a></li><li>DAU 与峰值群组数后付费价格详见 <a href="#">IM 计费说明</a></li><li>录制转码时长后付费价格详见 <a href="#">云端录制与投递计费说明</a> 和 <a href="#">云端混流转码计费说明</a></li><li>AI 智能识别时长后付费价格详见 <a href="#">AI 智能识别计费说明</a></li></ul> |                        |
| 支持平台          | iOS、Android、Web、Flutter、Electron（体验版和大规模直播版支持，单主播版和多人连麦版限时提供）                                                                                                                                                                                                                |                        |

#### ❗ 说明：

- 开通条件：**TUILiveKit 为 IM 和 TRTC 联合提供的服务，因此在开通 TUILiveKit 需首先保证应用已有对应版本的 IM 基础套餐包。可了解更多详情请 [单击购买](#)。
- 对应 TRTC 包月套餐：**TUILiveKit 为 IM 和 TRTC 联合提供的服务，同时 TUILiveKit 为 TRTC 包月套餐提供的功能项之一，因此购买 TUILiveKit 相当于订阅对应版本 TRTC 包月套餐，**仅收取一次费用**，更多 TRTC 包月套餐功能及版本说明可参见 [TRTC 版本功能](#)。示例：您在 IM 为专业版/旗舰版或企业版的情况下，购买了上表中价格为 9249 元/月的大规模直播版的版本，则：您订阅了 TRTC 旗舰版包月套餐，同时开通了 TUILiveKit 大规模直播版。
- 赠送音视频时长：**购买 TUILiveKit 相当于订阅对应版本 TRTC 包月套餐，TRTC 包月套餐将赠送音视频时长用量，有效期一个月，仅可用于抵扣订阅套餐服务的应用（SDKAppId）产生的实时音视频音视频时长用量，超额用量计费说明请参见 [音视频时长计费说明](#)。

## 计费说明

- 语音聊天室能力 TUILiveKit 由即时通信 IM 与实时音视频 TRTC 两个底层 PaaS 服务提供支持，**使用期间请务必保证您的 IM 应用和 TRTC 应用处于可用状态，否则语音聊天室相关功能将无法正常使用。**
- 语音聊天室能力购买后立即生效，体验版默认 7 天，付费版默认 1 个月。
- 语音聊天室能力暂不支持版本升降级。若您需要使用其他版本的语音聊天室能力，可在当前版本到期后重新购买，或在满足 [退款规则](#) 的前提下，退还当前版本语音聊天室能力，重新购买新的版本。
- 语音聊天室能力不支持更换 SDKAppId，您在选购时请务必确认您需要购买的 SDKAppId。

## 续费正式版

语音聊天室能力为 IM 增值服务之一，在进行续费时同样需要保证 IM 基础套餐包为专业版、旗舰版或企业版，因此续费时建议您同时对 IM 套餐包和语音聊天室能力进行续费，若您仅对语音聊天室能力进行续费，IM 套餐包到期后 IM 应用停用将导致语音聊天室能力无法正常使用。

具体操作如下：

1. 登录您的腾讯云 [费用中心](#)，在左侧导航中，选择 [续费管理](#)。
2. 分别为 IM 套餐包和音视频通话能力进行续费：
  - 右侧搜索框中搜索“IM + SDKAppID”，续费即时通信 IM 套餐包。
  - 右侧搜索框中搜索“TRTC + SDKAppID”，续费音视频通话能力。



3. 进入续费界面，选择续费时长，单击下一步。套餐包价格说明请参见 [价格总览](#)。



续费

×

已选中1个资源

| 资源ID/名称                          | 产品描述 | 当前到期时间     | 续费时长 <sup>①</sup> | 续费后到期时间 <sup>①</sup> | 单价 <sup>①</sup> | 续费金额         |
|----------------------------------|------|------------|-------------------|----------------------|-----------------|--------------|
| IM <div></div> <div>IM 旗舰版</div> | -    | 2025-04-21 | 1个月 ▾             | 2025-05-21           | <div>元/月</div>  | <div>元</div> |

自动续费

☒ 账户余额足够时，资源到期后按续费周期自动续费

协议条款

☒ 我已阅读并同意 [《自动续费规则》](#) 

优惠信息

当前暂无可用优惠券。

价格

元

 最终实付金额以支付订单时为准

下一步

取消

4. 在购买确认界面确认商品信息，提交订单并完成支付。

## 确认产品信息

**下单说明** 请确认产品信息后提交订单，如有优惠券可在支付时选择使用，**最终实付金额以支付订单时为准。**

## 产品清单

| ▼ 预付费产品 (1) |    |    |     |    |     |    | 实付金额 |
|-------------|----|----|-----|----|-----|----|------|
| 产品名称        | 配置 | 类型 | 单价  | 数量 | 时长  | 总价 | 订单金额 |
| 即时通信IM_续费   |    | 续费 | 元/月 | x1 | 1个月 | 元  | 元    |

## 选择优惠券

代金券 (0)

☒ 使用代金券抵扣 0.00元 兑换优惠券  
暂无可用代金券

实付金额 元

去支付

## 开启自动续期

由于 TUILiveKit 由实时音视频 TRTC 与即时通信 IM 联合提供服务，开启 TRTC 包月套餐和 IM 套餐包的自动续期后，TUILiveKit 的自动续期即可开启。您可以在购买页内下单时即勾选开启自动续期。

时长

1个月

2个月

3个月

4个月

5个月

6个月

更多

自动续费

☐ 账户余额足够时，到期后自动按月续费

协议条款

☐ 我已阅读并同意 [《实时音视频服务等级协议》](#) [《即时通信IM服务等级协议》](#)

若您已购买完成，可通过**控制台管理自动续费**和**费用中心管理自动续费** 2种方式管理自动续费，具体操作如下：

## 控制台管理自动续费

登录 [即时通信 IM 控制台](#) > [音视频服务 RTC](#)，在含 UI 低代码场景方案 > [直播/语聊房（TUILiveKit）](#) 功能区点击**开通自动续费**。

## 含 UI 低代码场景方案

音视频通话 (TUICallKit) 多人音视频 (TUIRoomKit) **在线直播/语音聊天室 (TUILiveKit)**TUILiveKit 是 IM 与 TRTC 联合打造的直播组件，适用于语音/视频直播或语聊房场景（参考微信视频号），查看 [场景方案](#) 介绍。

## 基本信息

服务状态 **已开通**

当前版本 **大规模直播版** [版本详情](#)

到期时间 2025-04-20 [续期指引](#)

自动续费 **未开启** [开启自动续费](#)

## 快速接入

10 分钟快速跑通代码，几行代码将方案正式集成到您的项目中

[Android 集成文档](#)

[iOS 集成文档](#)

[Flutter 集成文档](#)

[Electron 集成文档](#)

## Demo 体验

在线直播、语音聊天室



Android Demo



iOS Demo

## 费用中心管理自动续费

1. 登录您的腾讯云 [费用中心](#)，在左侧导航中，选择 [续费管理](#)。
2. 将 TRTC 包月套餐和 IM 套餐包设置为自动续费：
  - 右侧搜索框中搜索“IM + SDKAppID”，将即时通信 IM 套餐包设为自动续费。
  - 产品类别选择实时音视频，右侧搜索框中搜索“TRTC + SDKAppID”，将 TRTC 包月套餐设为自动续费。

续费管理

续费价格受实时报价和优惠变动的影响，可能会与之前新购、续费、导出的续费清单价格不一致，请以实时续费支付页显示价格为准。

全部到期 已到期 7天内到期 15天内到期 **30天内到期** 2025-03-21 ~ 2025-04-20 全部地域 运行中

搜索产品

全部产品 21

实时通信IM 4

实时音视频 6

音视频传输 SDK (腾讯云原生) 1

媒体处理MPS 10

手动续费 4 自动续费 2 到期不续 0

批量续费 设为自动续费 设为到期不续 统一到期日

| 资源ID/名称 | 子产品名称    | 产品描述 | 地域/可用区         | 所属项目 | 资源状态 | 到期倒计时 | 开始时间                | 单价                  | 操作                           |
|---------|----------|------|----------------|------|------|-------|---------------------|---------------------|------------------------------|
| TRTC    | TRTC包月套餐 |      | 华东地区 (广州) 广州一区 | 默认项目 | 运行中  | 剩余 4天 | 2025-02-26 17:02:59 | 2025-03-26 17:02:59 | 续费 <a href="#">设为自动续费</a> 更多 |

- 产品类别选择即时通信，右侧搜索框中搜索“IM + SDKAppID”，将即时通信 IM 套餐包设为自动续费。

费用中心

[① 订单管理](#)
[续费管理](#)
[资源包续费](#)
[② 一键续费](#)
[续费设置](#)
[回到旧版](#)

续费管理

续费价格受实时报价和优惠变动的影响，可能会与之前新购、续费、导出的续费清单价格不一致，请以实时续费支付页显示价格为准。

[全部到期](#)
[已到期](#)
[7天内到期](#)
[15天内到期](#)
[30天内到期](#)
[选择日期](#)
[选择日期](#)
[全部地域](#)
[运行中](#)

搜索产品

全部产品 1952

[云联络中心](#) 102
 [即时通信IM](#) 1058
 [实时音视频-低代码开发平台](#) 4
 [虚拟直播](#) 2
 [实时音视频](#) 331
 [SSL证书](#) 26
 [音视频转码 SDK \(腾讯云视立方\)](#) 374
 [媒体处理MPS](#) 55

手动续费 1330 自动续费 337 到期不续 285

[批量续费](#)
[设为自动续费](#)
[设为到期不续](#)
[统一到期日](#)

资源ID名称

产品名称

子产品名称

产品描述

地域/可用区

所属项目

资源状态

到期倒计时

开始到期时间

单价

操作

IM 旗舰版

即时通信IM

即时通信IM\_旗舰版

其他地区 (其他)

默认项目

运行中

31天

2025-03-21 15:38:22

2025-04-21 15:38:22

¥12/月

续费 [设为自动续费](#) [更多](#)

### 3. 操作完成您可登录 [即时通信 IM 控制台](#)，查看自动续期开启状态。

含 UI 低代码场景方案

[音视频通话 \(TUICallKit\)](#)
[多人音视频 \(TUIRoomKit\)](#)
[在线直播/语音聊天室\(TUILiveKit\)](#)

TUILiveKit 是 IM 与 TRTC 联合打造的直播组件，适用于语音/视频直播或语聊房场景（参考微信视频号），查看 [场景方案](#) 介绍。

基本信息

服务状态

已开通

当前版本

大规模直播版 [版本详情](#)

到期时间

2025-04-20 [续期指引](#)

自动续费

已开启 [①](#)

快速接入

10 分钟快速跑通代码，几行代码将方案正式集成到您的项目中

[Android 集成文档](#)
[iOS 集成文档](#)
[Flutter 集成文档](#)
[Electron 集成文档](#)

Demo 体验

在线直播、语音聊天室

Android Demo

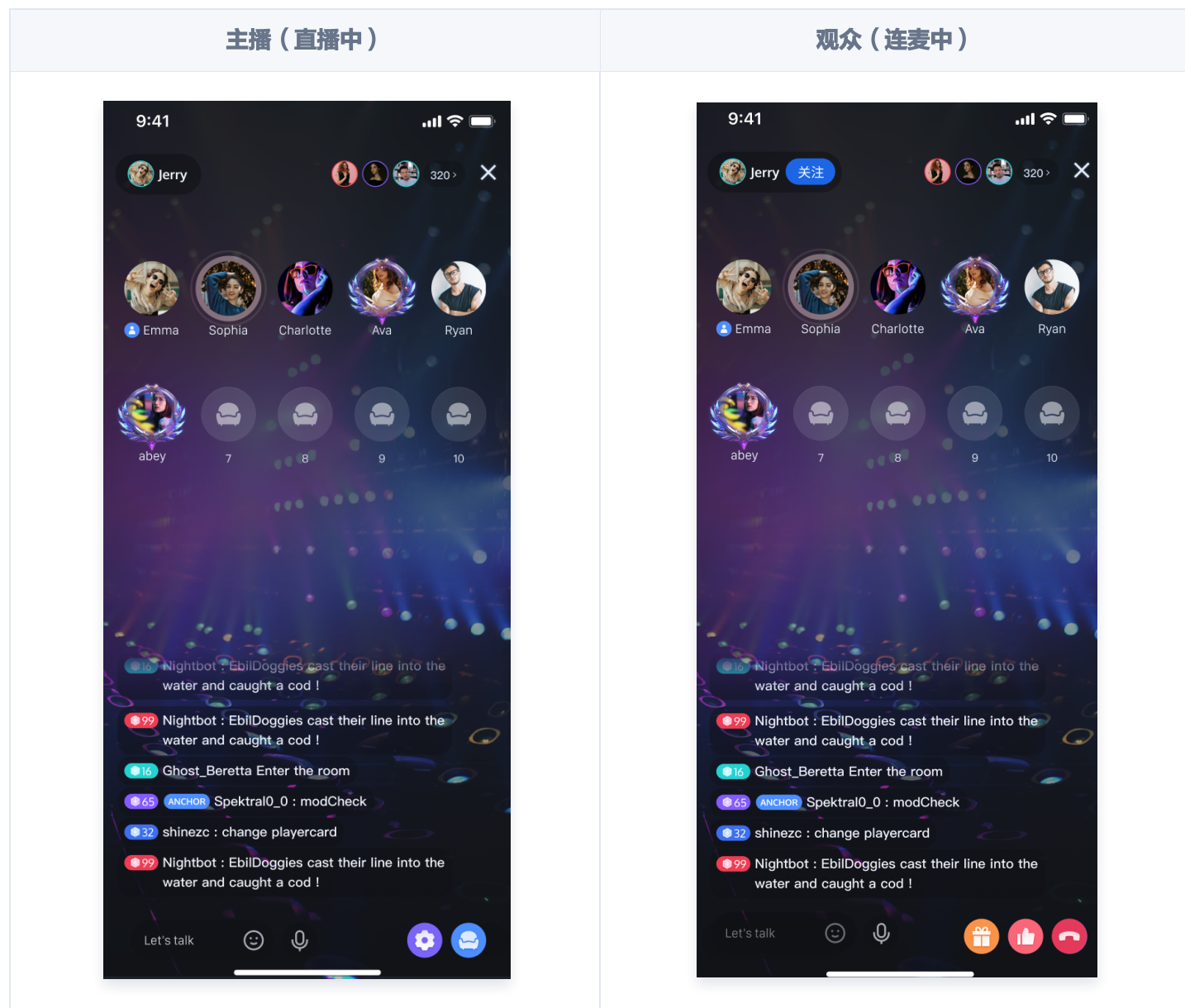
iOS Demo

# 跑通 Demo (TUILiveKit)

## iOS

最近更新时间: 2025-04-07 17:43:42

本文将为您介绍如何快速跑通语音聊天室 Demo。跟随本文档，您可以在 10 分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的语音直播功能。



## 环境准备

- Xcode 15 及以上。
- iOS 13.0 及以上。
- CocoaPods 环境安装, 请点击 [查看](#)。

- 如果您的接入和使用中遇到问题，请参见 [常见问题](#)。

## 步骤一：下载 Demo

1. 从 github 下载 [TUILiveKit Demo](#) 源码，或者直接在命令行运行以下命令：

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

2. 在命令行中进入 iOS 项目目录：

```
cd TUILiveKit/iOS/Example
```

3. 加载依赖库：

```
pod install
```

### ⚠ 注意：

如果您尚未安装 CocoaPods，可以参见 [此处](#) 了解如何安装。

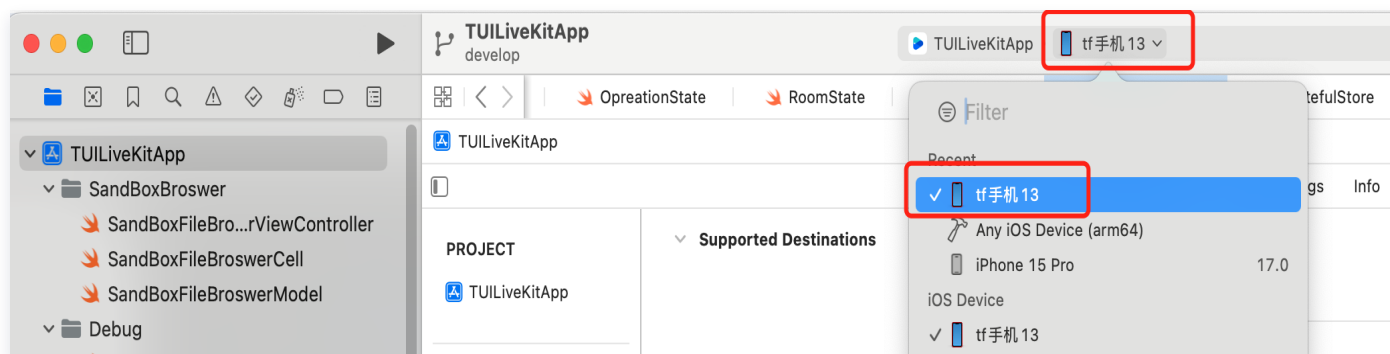
## 步骤二：配置 Demo

1. 请参见 [开通服务（TUILiveKit）](#)，领取体验版或者开通付费版。申请成功后即可获得14天的体验版，可免费体验 TUILiveKit 的全部功能，并获取 SDKAppID 和 SDKSecretKey。
2. 打开 `/iOS/Example/Debug/GenerateTestUserSig.swift` 文件，将获取到的对应的 SDKAppID 和 SDKSecretKey 填入其中：

```
TUICallKitApp > Debug > GenerateTestUserSig > No Selection
10 import CommonCrypto
11 import zlib
12
13 /**
14  * Tencent Cloud SDKAppId, which needs to be replaced with the SDKAppId under your own account.
15  *
16  * Enter Tencent Cloud IM to create an application, and you can see the SDKAppId, which is the unique identifier used by Tencent Cloud to
17  */
18 let SDKAPPID: Int = 0
19
20 /**
21  * Signature expiration time, it is recommended not to set it too short
22  *
23  * Time unit: seconds
24  * Default time: 7 x 24 x 60 x 60 = 604800 = 7 days
25  */
26 let EXPIRETIME: Int = 604_800
27
28 /**
29  * Encryption key used for calculating the signature, the steps to obtain it are as follows:
30  *
31  * step1. Enter Tencent Cloud IM, if you do not have an application yet, create one,
32  * step2. Click "Application Configuration" to enter the basic configuration page, and further find the "Account System Integration" section.
33  * step3. Click the "View Key" button, you can see the encryption key used to calculate UserSig, please copy and paste it into the following
34  *
35  * Note: This solution is only applicable to debugging demos.
36  * Before going online officially, please migrate the UserSig calculation code and keys to your backend server to avoid traffic theft caused by
37  */
38 let SECRETKEY = ""
39
```

## 步骤三：跑通 Demo

1. 在 XCode 如下图所示处选择您要将 Demo 运行的设备：



2. 选择完成后，点击运行将我们的 Demo 运行到目标设备上。

3. 当 Demo 成功运行到设备上后，您可以通过如下步骤来开启语音直播和加入语音直播。

## 步骤四：开始直播

### ⚠ 注意：

为了使您可以体验完整的语音直播流程，请将 Demo 分别在两台设备上登录两个用户，一方作为主播，一方作为观众。

1. 登录 & 注册

请您在 用户ID 处输入的 ID 。如果您当前的 UserID 未曾被使用过，此时会进入到注册界面，您可以在该界面为自己设置头像及昵称。

| 主播端 ( mike ) |  | 观众端 ( vince ) |  |
|--------------|--|---------------|--|
|              |  |               |  |

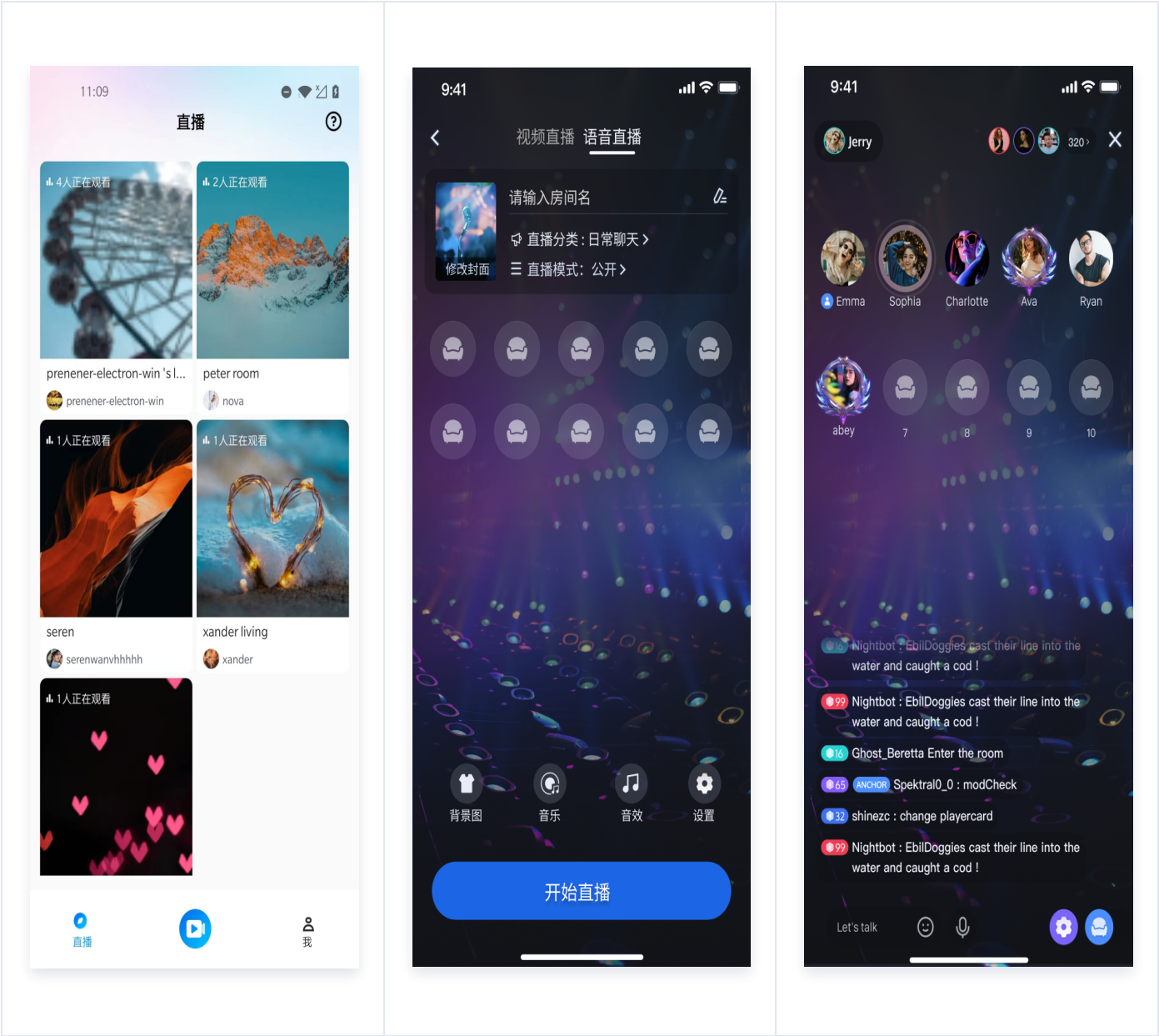
**注意：**  
尽量避免使您的 UserID 被设置成 “1”、“123”、“111” 等简单字符串，由于 TRTC 不支持同一个 UserID 多端登录，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。

2. 主播开启直播

点击主页底部中间的“开播”按钮，即可进入开播预览页面，再单击开始直播即可开始直播了。

| 主播（进房前） | 主播（预览） | 主播（直播中） |
|---------|--------|---------|
|---------|--------|---------|

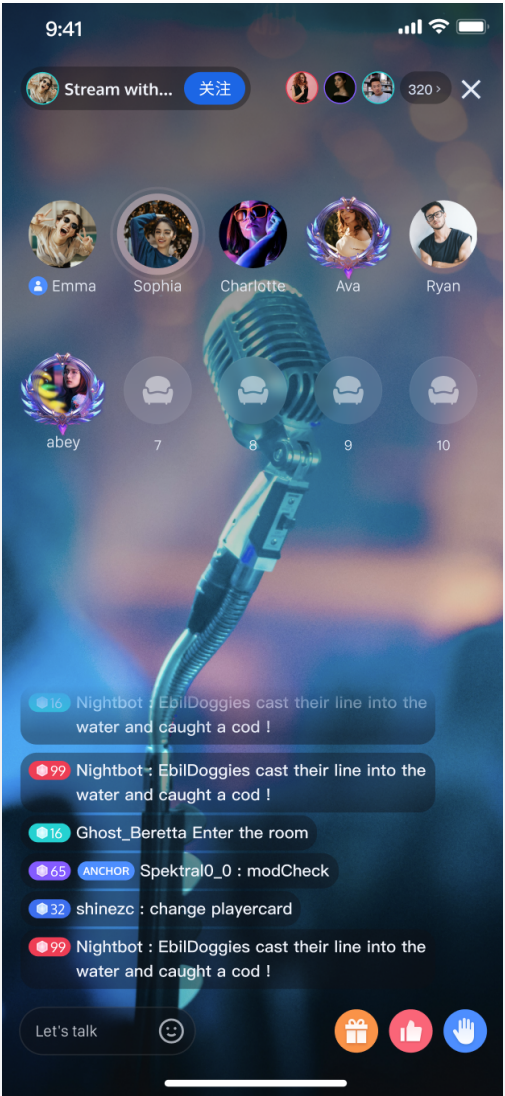
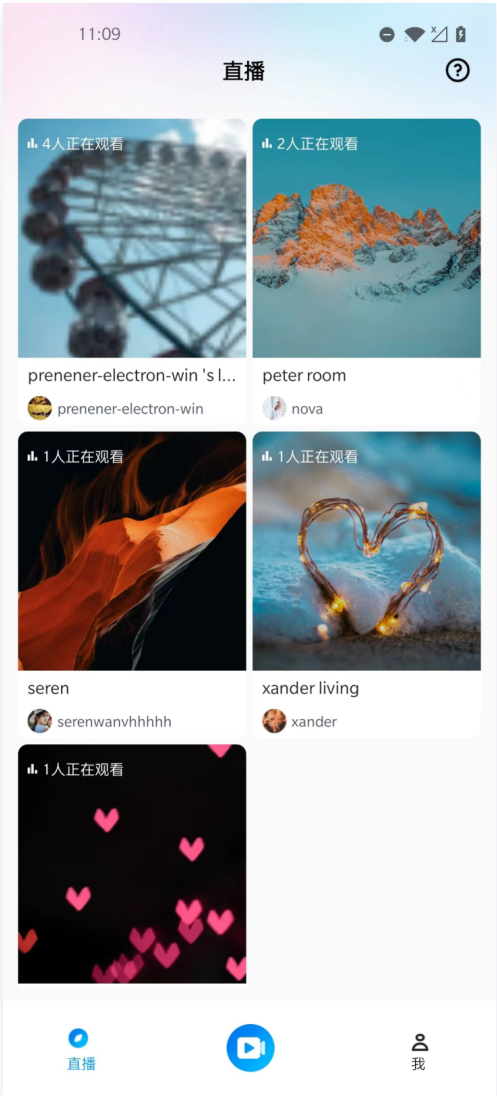




3. 观众加入直播间

点击直播列表里任意一个房间，即可进入直播间。

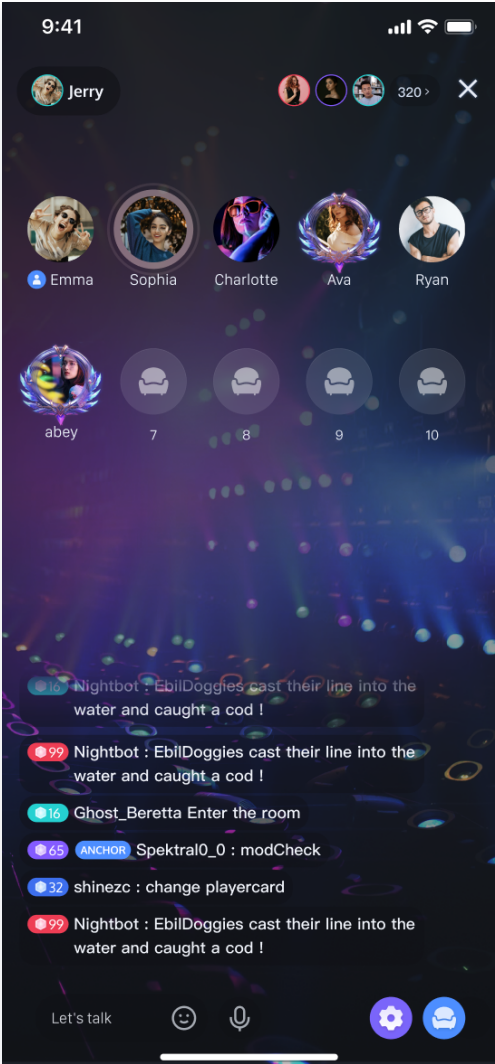
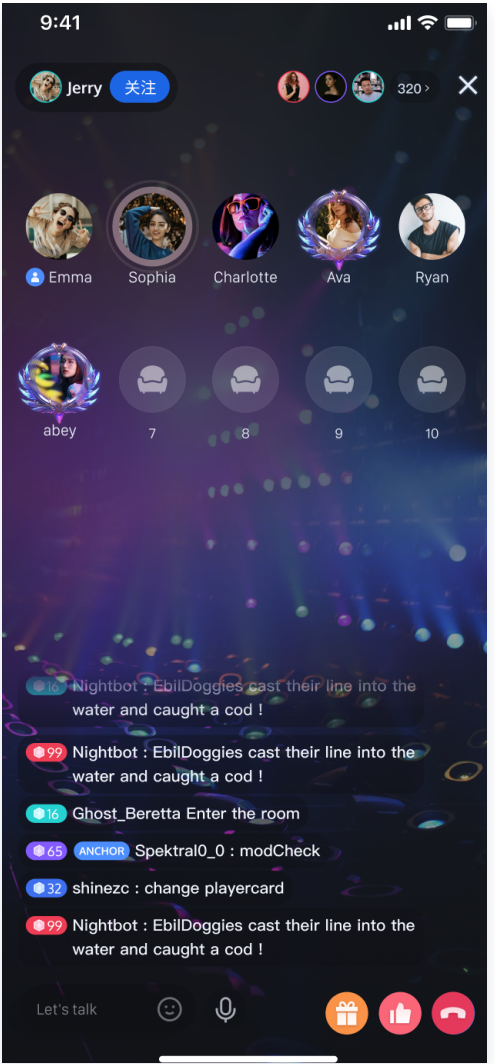
| 观众（进房前） | 观众（进房后） |
|---------|---------|
|---------|---------|



# Android

最近更新时间：2025-04-07 15:22:32

本文将为您介绍如何快速跑通语音直播 Demo。跟随本文档，您可以在 10 分钟内跑通 Demo，并最终体验一个包含完备 UI 界面的语音直播功能。

| 主播（直播中）                                                                            | 观众（连麦中）                                                                             |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|  |  |

## 环境准备

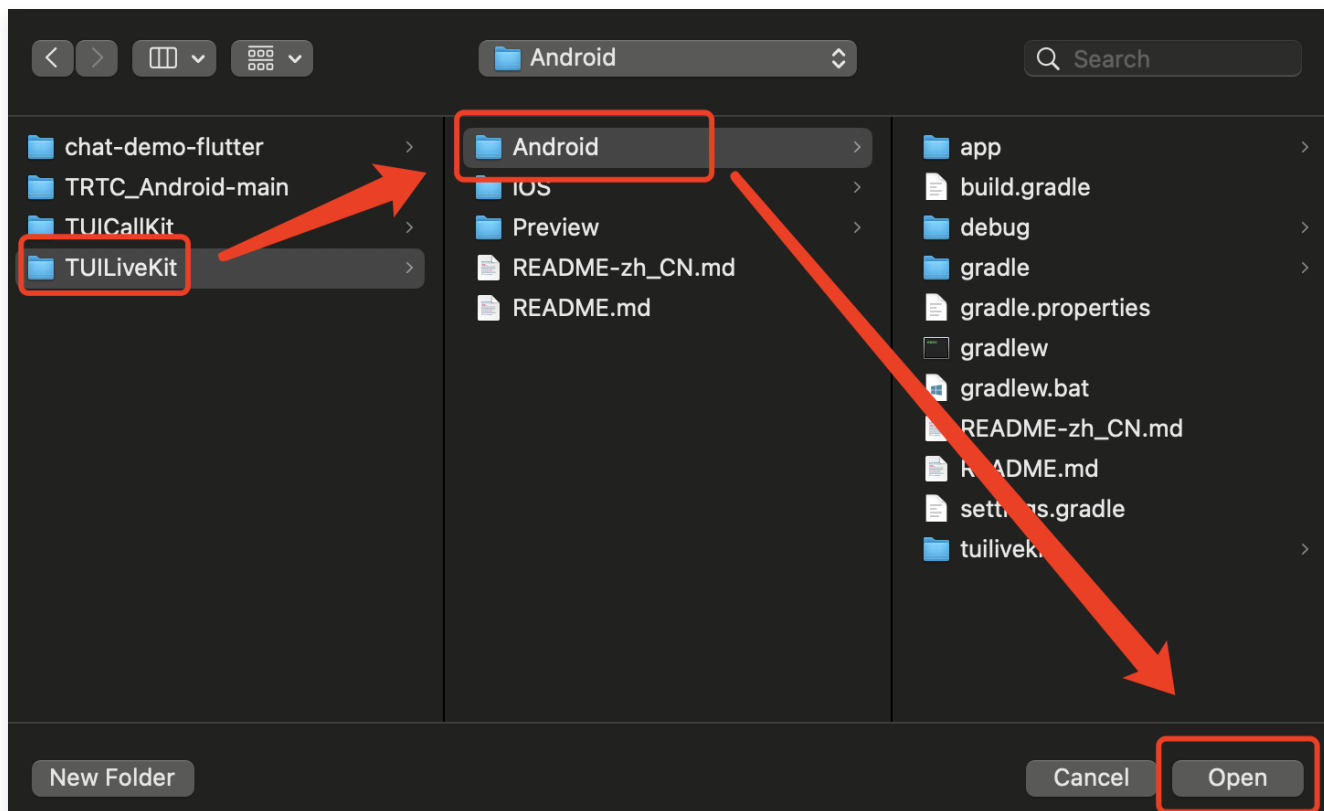
- Android 5.0 (SDK API level 21)及以上。
- Gradle 4.2.1 及以上。
- 两台 Android 5.0 以上的设备。

## 步骤一：下载 Demo

1. 从 github 下载 [TUILiveKit Demo](#) 源码，或者直接在命令行运行以下命令：

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

2. 通过 Android Studio 打开 TUILiveKit Android 项目：

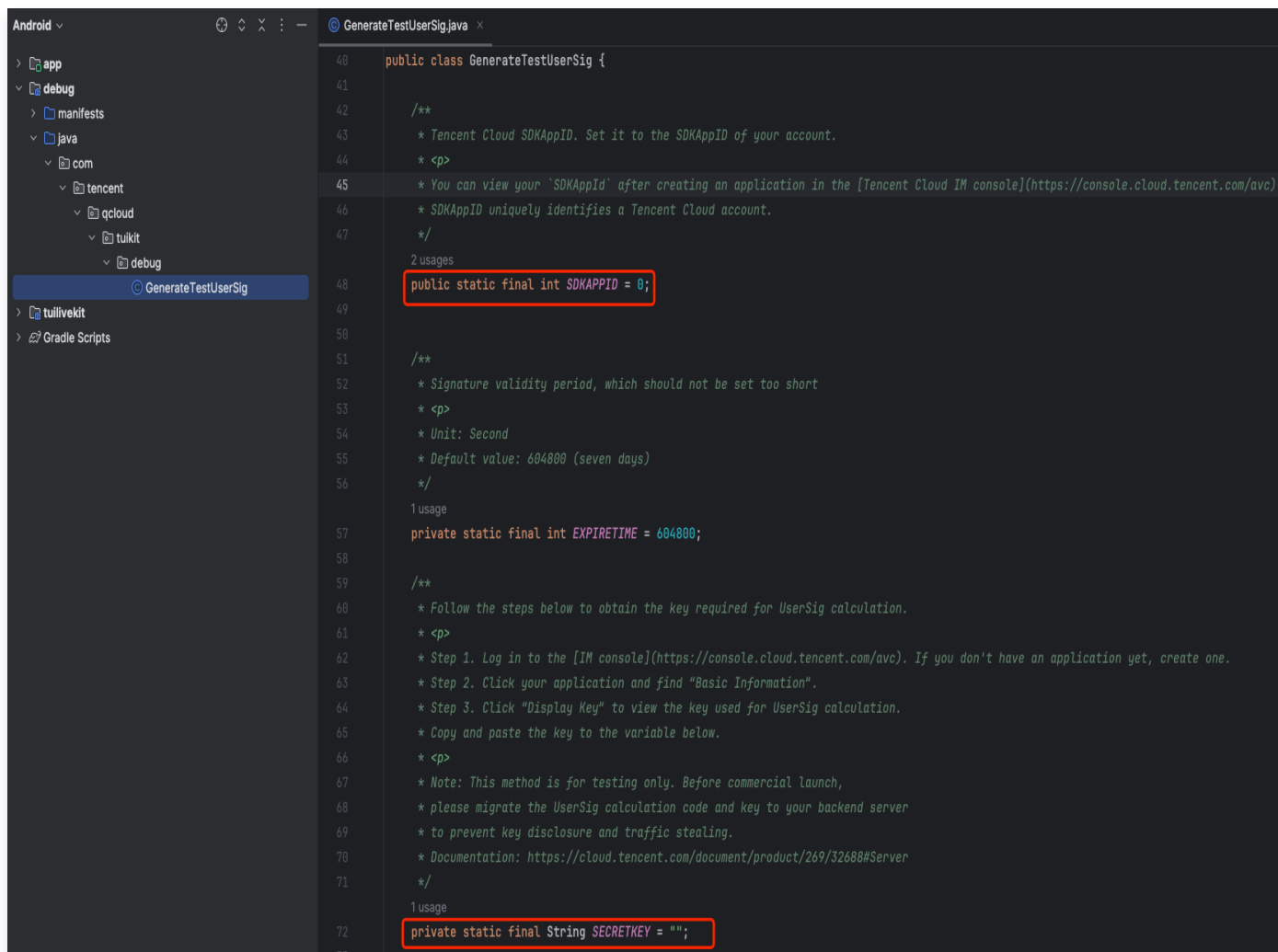


## 步骤二：配置 Demo

1. 请参见 [开通服务（TUILiveKit）](#)，领取体验版或者开通付费版。成功后即可获得14天的体验版，可免费体验 TUILiveKit 的全部功能，并获取 SDKAppID 和 SDKSecretKey。
2. 打开

```
Android/debug/src/main/java/com/tencent/qcloud/tuikit/debug/GenerateTestUserSig.java
```

文件，将获取到对应的 SDKAppID 和 SDKSecretKey 填入其中：



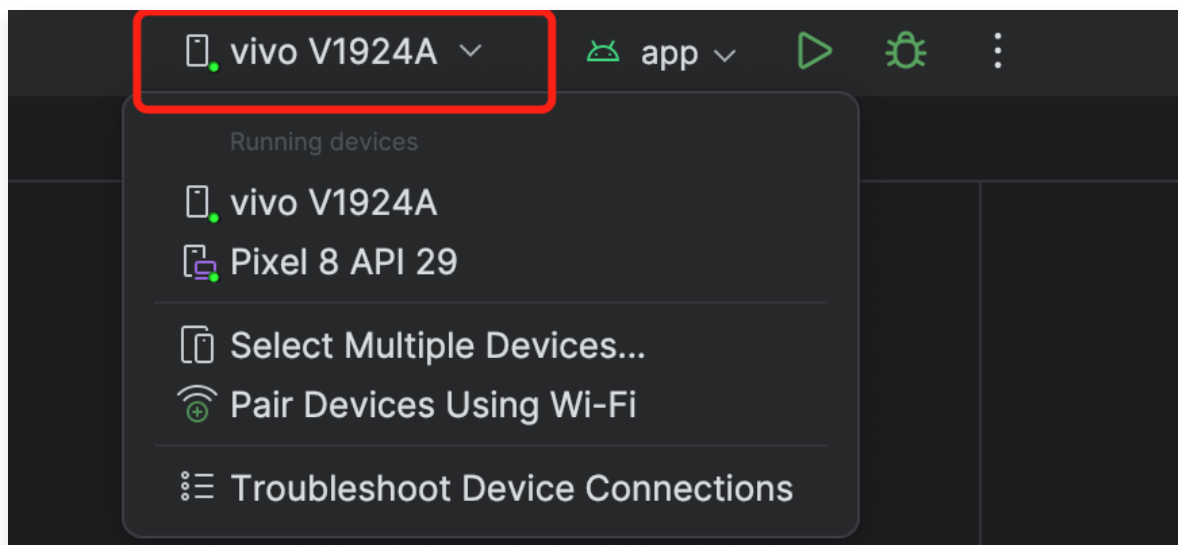
```

40 public class GenerateTestUserSig {
41
42     /**
43      * Tencent Cloud SDKAppID. Set it to the SDKAppID of your account.
44      * <p>
45      * You can view your 'SDKAppID' after creating an application in the [Tencent Cloud IM console](https://console.cloud.tencent.com/avc).
46      * SDKAppID uniquely identifies a Tencent Cloud account.
47      */
48     public static final int SDKAPPID = 0;
49
50     /**
51      * Signature validity period, which should not be set too short
52      * <p>
53      * Unit: Second
54      * Default value: 604800 (seven days)
55      */
56     private static final int EXPIRETIME = 604800;
57
58     /**
59      * Follow the steps below to obtain the key required for UserSig calculation.
60      * <p>
61      * Step 1. Log in to the [IM console](https://console.cloud.tencent.com/avc). If you don't have an application yet, create one.
62      * Step 2. Click your application and find "Basic Information".
63      * Step 3. Click "Display Key" to view the key used for UserSig calculation.
64      * Copy and paste the key to the variable below.
65      * <p>
66      * Note: This method is for testing only. Before commercial launch,
67      * please migrate the UserSig calculation code and key to your backend server
68      * to prevent key disclosure and traffic stealing.
69      * Documentation: https://cloud.tencent.com/document/product/269/32688#Server
70      */
71     private static final String SECRETKEY = "";
72
73 }

```

## 步骤三：跑通 Demo

1. 在 Android Studio 右上角如下图所示处选择您要将 Demo 运行的设备：





2. 选择完成后点击运行，将 TUILiveKit Android Demo 运行到目标设备上。



3. 当 Demo 成功运行到设备上后，您可以通过如下步骤来发起直播和观看直播。

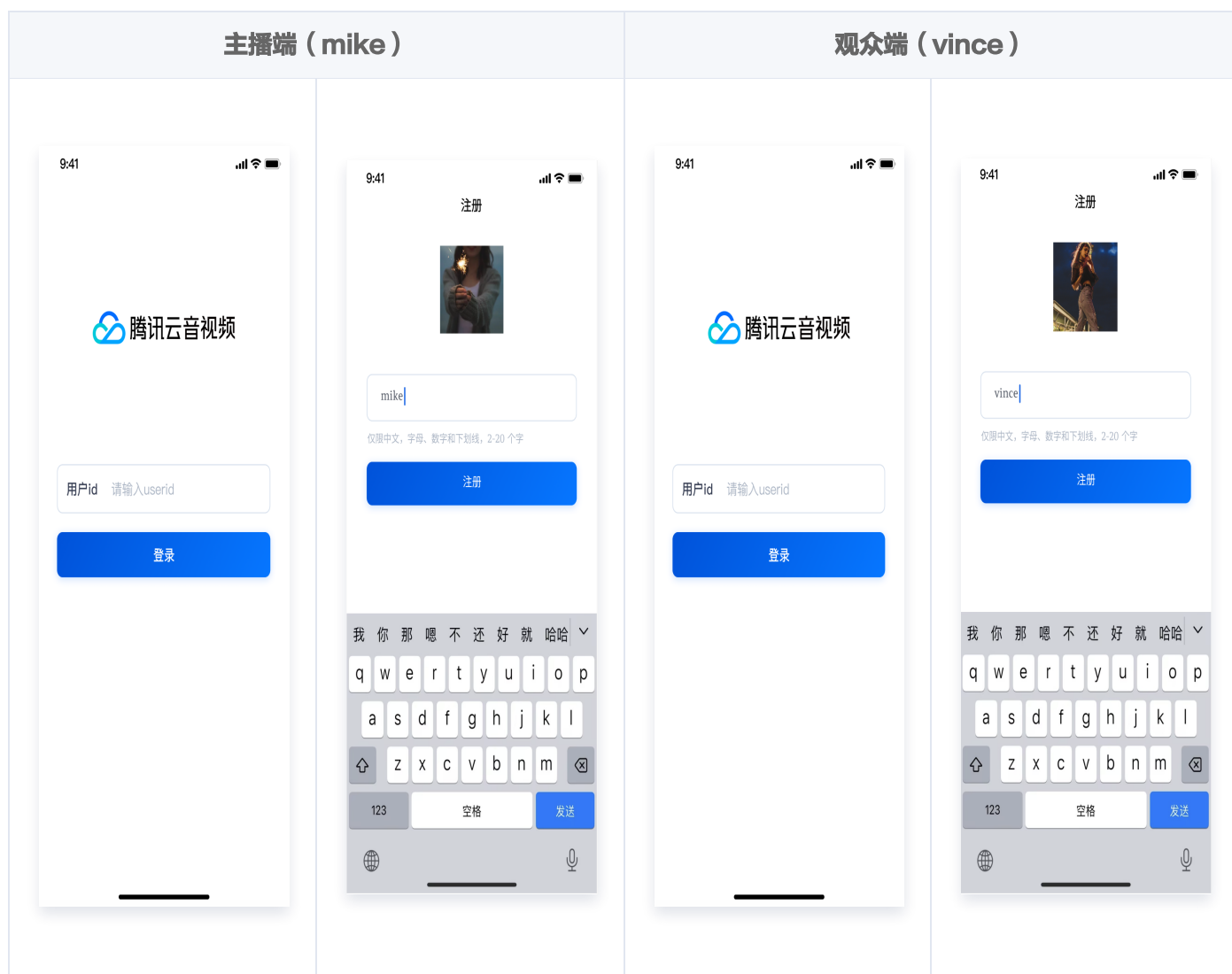
## 步骤四：开始直播

### ⚠ 注意：

为了使您可以体验完整的语音直播流程，请将 Demo 分别在两台设备上登录两个用户，一方作为主播，一方作为观众。

### 1. 登录 & 注册

请您在 **用户ID** 处输入的 ID。如果您当前的 UserID 未曾被使用过，此时会进入到注册界面，您可以在该界面为自己设置头像及昵称。

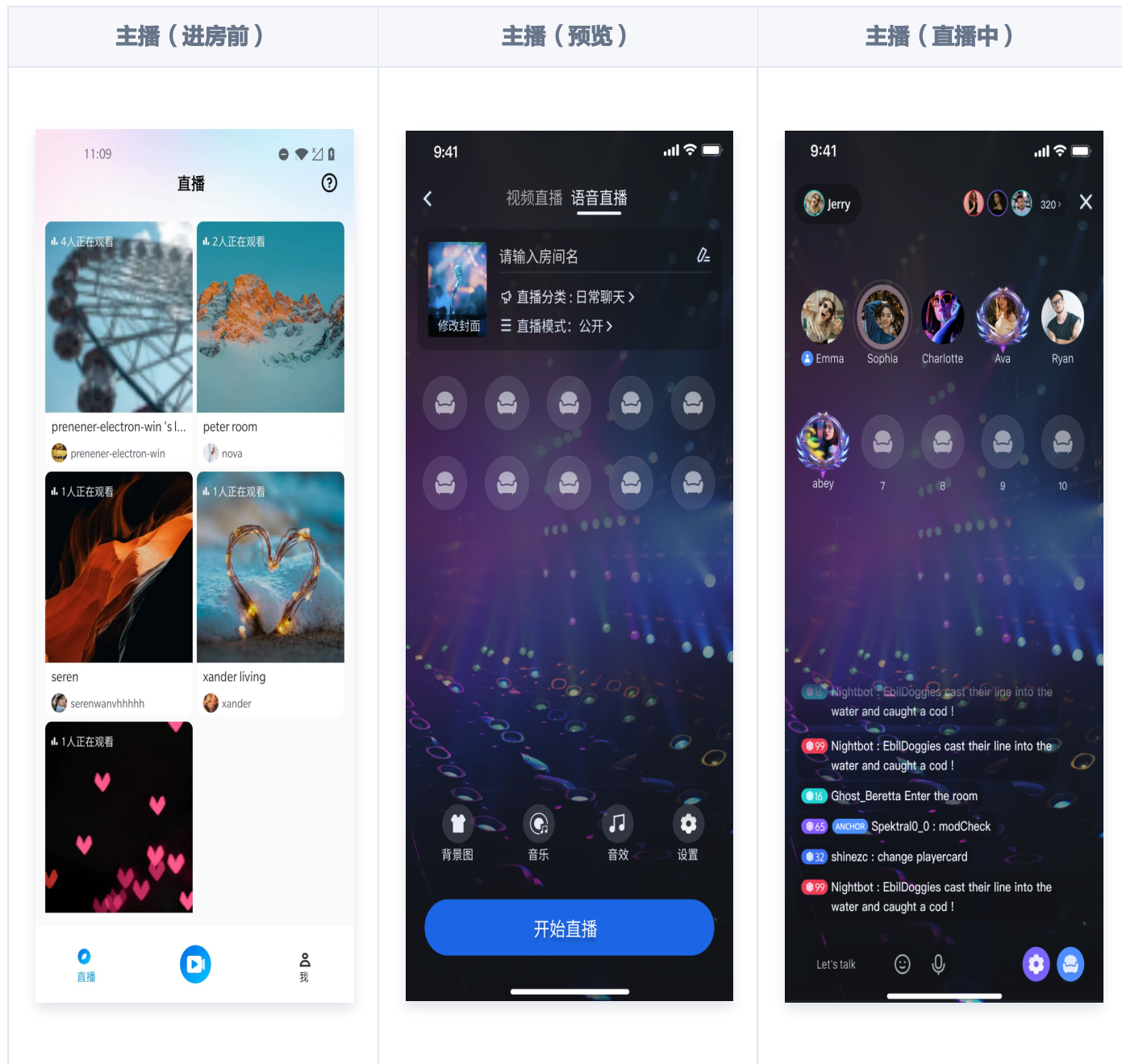


**⚠ 注意:**

尽量避免使您的 UserID 被设置成“1”、“123”、“111”等简单字符串，由于 TRTC 不支持同一个 UserID 多端登录，所以在多人协作开发时，形如“1”、“123”、“111”这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。

## 2. 主播开启直播

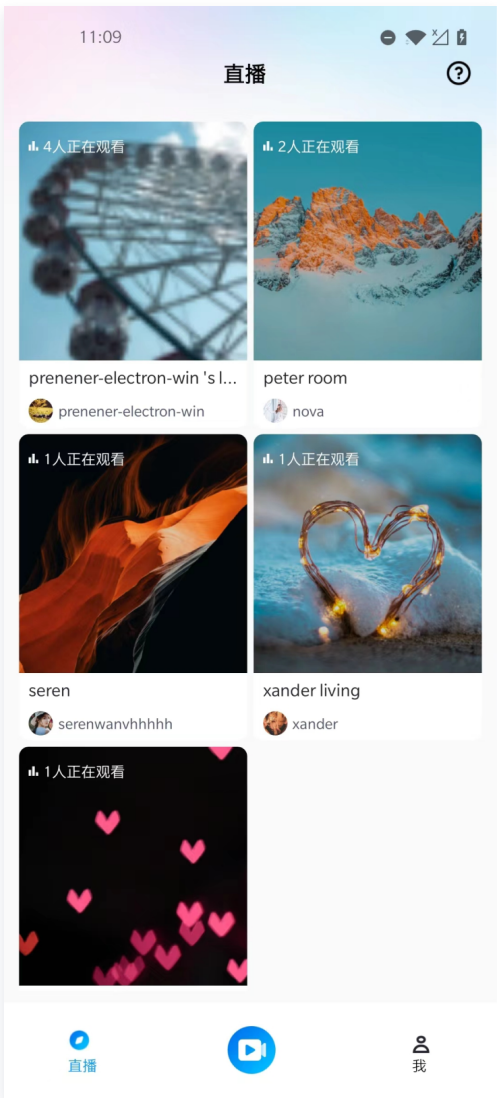
点击主页底部中间的“开播”按钮，即可进入开播预览页面，再单击**开始直播**即可开始直播了。



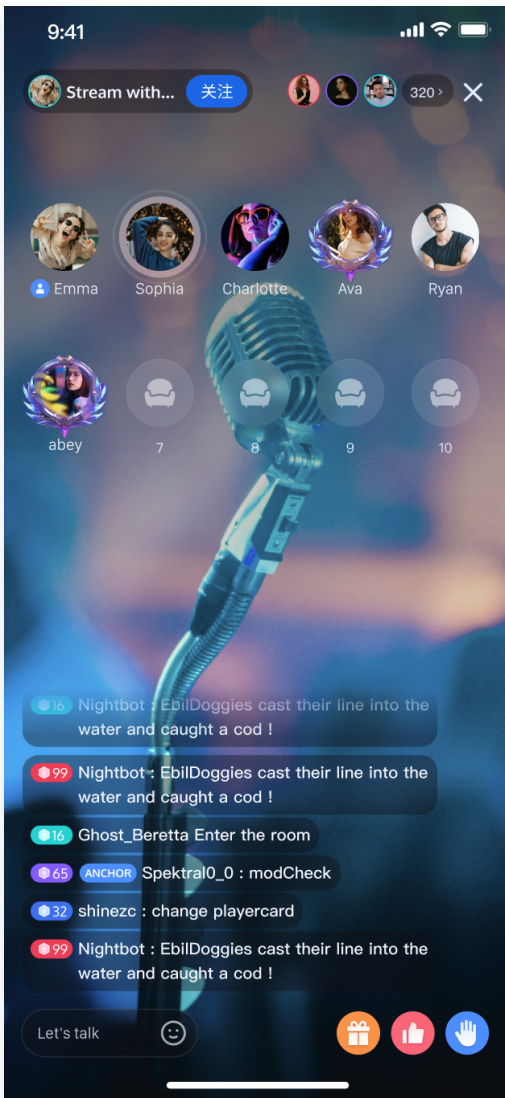
## 3. 观众加入直播间

点击直播列表里任意一个房间，即可进入直播间。

观众（进房前）



观众（进房后）





# 快速接入（TUILiveKit） Android

最近更新时间：2025-07-29 14:59:11

本文将介绍如何在短时间内完成 TUILiveKit 组件的接入，跟随本文档，您将在一小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的语音聊天室功能。

| 主播 | 观众 |
|----|----|
|    |    |

## 环境准备

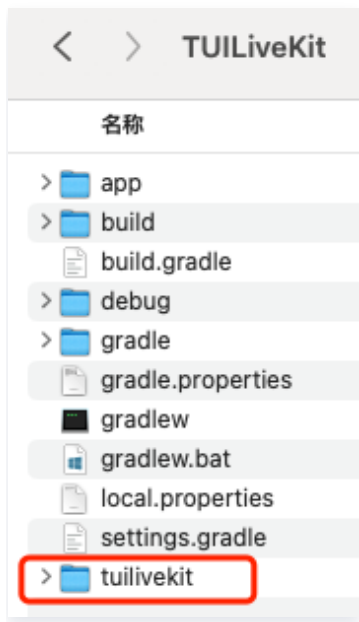
- Android 5.0（SDK API Level 21）及以上版本。
- Gradle 7.0 及以上的版本。
- Android 5.0 及以上的手机设备。

## 步骤1：开通服务

请参见 [开通服务（TUILiveKit）](#)，领取体验版或者开通付费版。

## 步骤2：下载 TUILiveKit 组件

在 [GitHub](#) 中克隆/下载代码，然后拷贝 Android 目录下的 tuilivekit 子目录到您当前工程中的 app 同一级目录中，如下图：



## 步骤3：工程配置

1. 在工程根目录下的 `settings.gradle.kts`（或 `settings.gradle`）文件中添加 `jitpack` 仓库依赖（下载播放礼物 `svg` 动画的三方库 `SVGAPlayer`）：

### settings.gradle.kts

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

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

### settings.gradle

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

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

2. 在工程根目录的 `settings.gradle.kts` (或`settings.gradle`) 文件中增加如下代码，它的作用是将 [步骤二](#) 中下载的 `tuilivekit` 组件导入到您当前的项目中：

#### settings.gradle.kts

```
include(":tuilivekit")
```

#### settings.gradle

```
include ':tuilivekit'
```

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

#### build.gradle.kts

```
api(project(":tuilivekit"))
```

#### build.gradle

```
api project(':tuilivekit')
```

#### ❗ 说明：

TUILiveKit 工程内部已经默认依赖：`TRTC SDK`、`IM SDK`、`tuiroomengine` 以及公共库 `tuicore`，不需要开发者单独配置。如需进行版本升级，则修改 `tuilivekit/build.gradle` 文件即可。

4. 由于我们在 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在

`proguard-rules.pro` 文件中添加如下代码：

```
-keep class com.tencent.** { *; }
-keep class com.trtc.uikit.livekit.livestreamcore.** { *; }
-keep class com.trtc.uikit.livekit.component.gift.store.model.** { *; }
-keep class com.squareup.wire.** { *; }
-keep class com.opensource.svgaplayer.proto.** { *; }

-keep class com.tcmediax.** { *; }
-keep class com.tencent.** { *; }
-keep class com.tencent.xmagic.** { *; }
-keep class androidx.exifinterface.** { *; }
-keep class com.google.gson.** { *; }
# Tencent Effect SDK - beauty
-keep class com.tencent.xmagic.** { *; }
-keep class org.light.** { *; }
-keep class org.libpag.** { *; }
-keep class org.extra.** { *; }
-keep class com.gyailib.** { *; }
-keep class com.tencent.cloud.iai.lib.** { *; }
-keep class com.tencent.beacon.** { *; }
-keep class com.tencent.qimei.** { *; }
-keep class androidx.exifinterface.** { *; }
```

5. 在 `app` 目录下找到 `AndroidManifest.xml` 文件，在 `application` 节点中添加

`tools:replace="android:allowBackup"`，覆盖组件内的设置，使用自己的设置。

```
// app/src/main/AndroidManifest.xml

<application
    ...

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

## 步骤4：登录

在您的项目中添加如下代码，它的作用是通过调用 `TUICore` 中的相关接口完成 `TUI` 组件的登录。这一步骤至关重要，只有在成功登录之后，您才能正常使用 `TUILiveKit` 提供的各项功能。

## Kotlin

```
//登录
TUILogin.login(applicationContext,
    1400000001, // 请替换为步骤一取到的 SDKAppID
    "denny", // 请替换为您的 UserID
    "xxxxxxxxxxxx", // 您可以在控制台中计算一个 UserSig 并填在这个位置
    object : TUICallback() {
        override fun onSuccess() {
            Log.i(TAG, "login success")
        }

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

## Java

```
//登录
TUILogin.login(context,
    1400000001, // 请替换为步骤一取到的 SDKAppID
    "denny", // 请替换为您的 UserID
    "xxxxxxxxxxxx", // 您可以在控制台中计算一个 UserSig 并填在这个位置
    new TUICallback() {
        @Override
        public void onSuccess() {
            Log.i(TAG, "login success");
        }

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

### 参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

| 参数       | 类型     | 说明                                                                                                                                                                                                                  |
|----------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SDKAppID | int    | 在步骤一中的最后一步中您已经获取到，这里不再赘述。                                                                                                                                                                                           |
| UserID   | String | 当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符和下划线。                                                                                                                                                                |
| userSig  | String | 使用 <a href="#">步骤一</a> 的第3步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 <a href="#">辅助工具</a> 生成一个临时可用的 UserSig。更多信息请参见 <a href="#">如何计算及使用 UserSig</a> 。 |

#### ❗ 说明：

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

## 步骤5：创建语音聊天室

在上述登录方法调用返回成功后，调用 [VoiceRoomKit](#) 的 [createRoom](#) 方法，创建您的语聊房直播间。

### Kotlin

```
import com.trtc.ukit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(applicationContext).createRoom("roomId",
VoiceRoomDefine.CreateRoomParams())
```

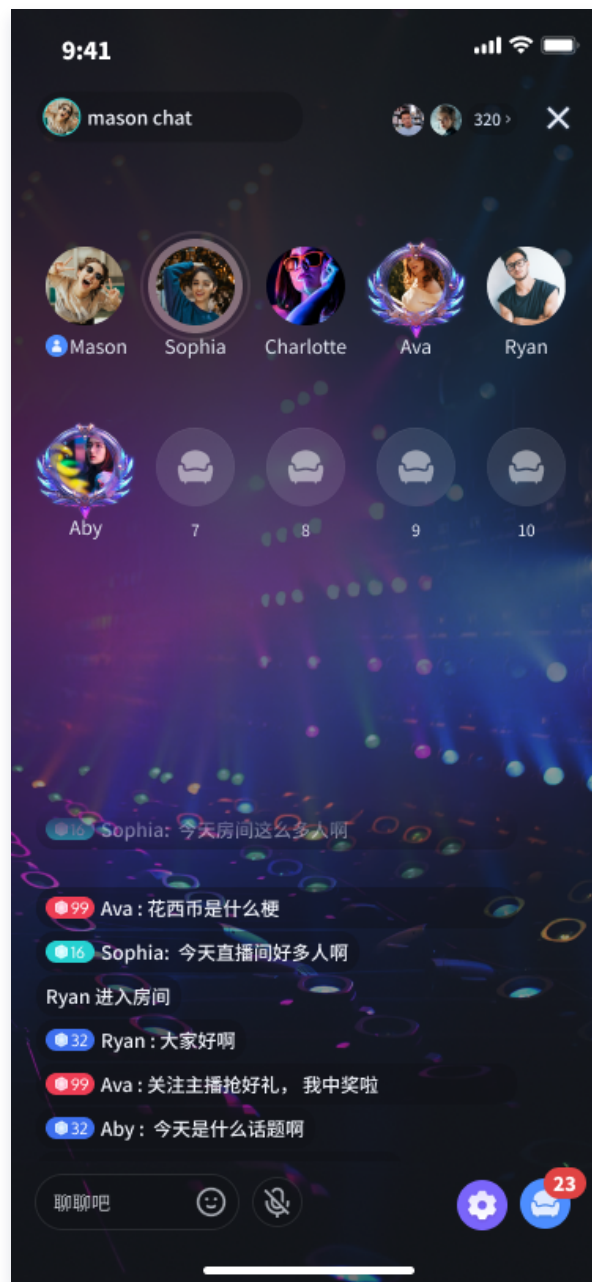
### Java

```
import com.trtc.ukit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(getApplicationContext()).createRoom("roomId",
new VoiceRoomDefine.CreateRoomParams());
```

语音聊天室预览画面

语音聊天室房间内画面



## 步骤6：加入语音聊天室

在上述登录方法调用返回成功后，调用 `VoiceRoomKit` 的 `enterRoom` 方法，指定房间 ID，进入语聊房直播间。

### Kotlin

```
import com.trtc.ukit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(applicationContext).enterRoom("roomId")
```

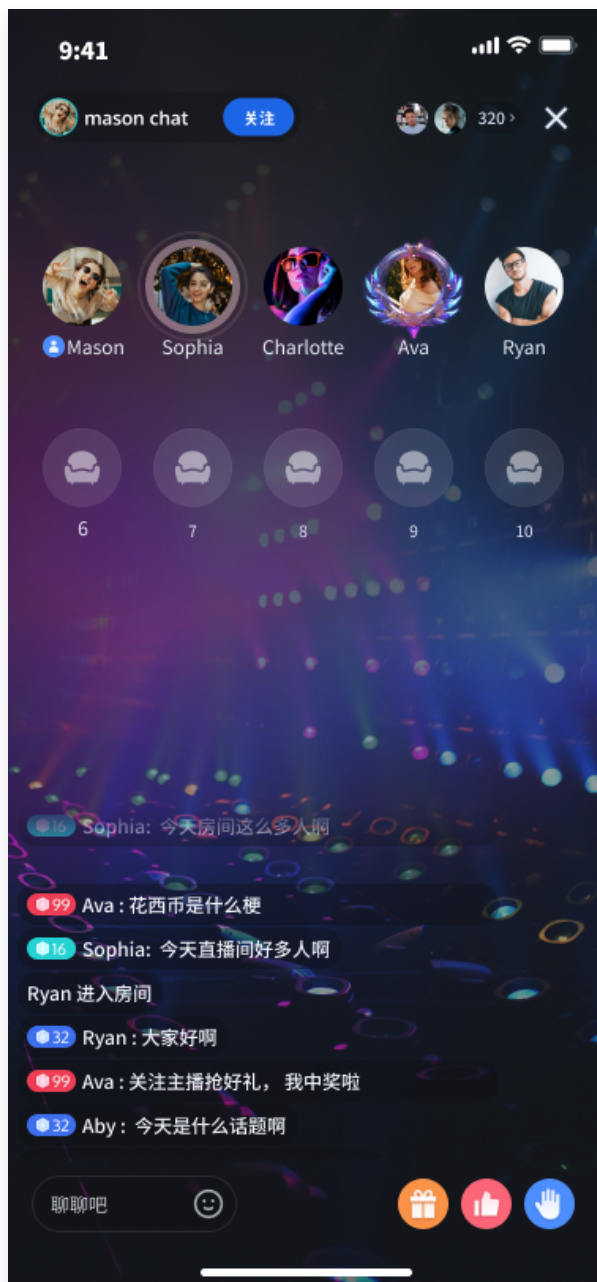


## Java

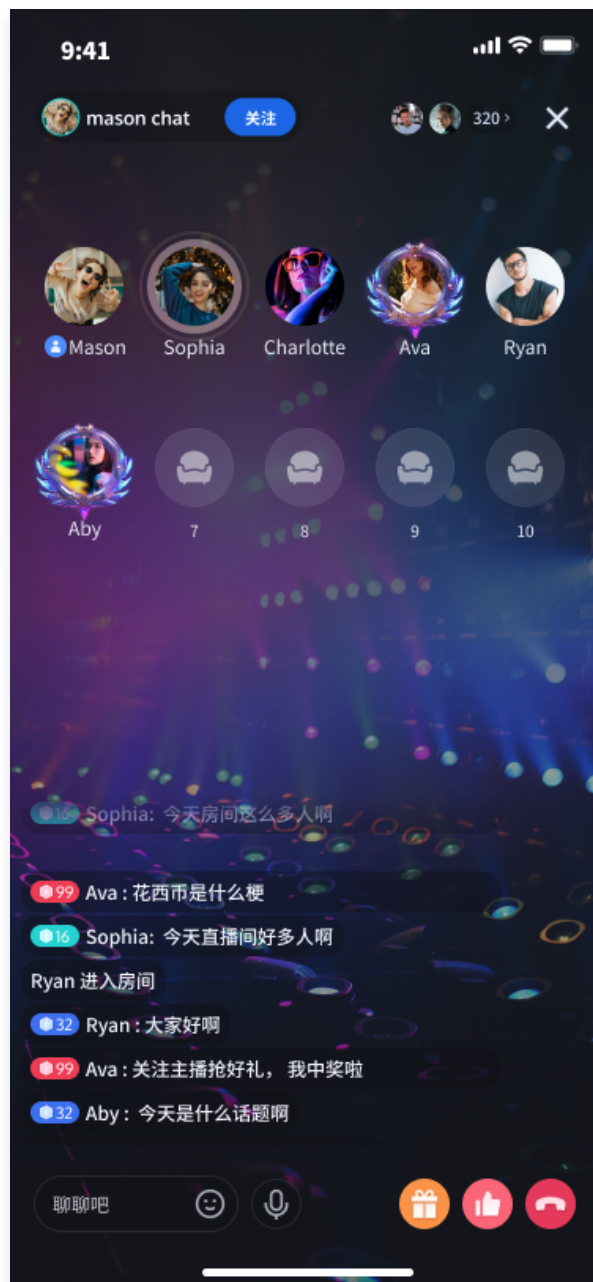
```
import com.trtc.ukit.livekit.VoiceRoomKit;

VoiceRoomKit.createInstance(getApplicationContext()).enterRoom("roomId");
```

### 语音聊天室



### 语音聊天室



## 常见问题

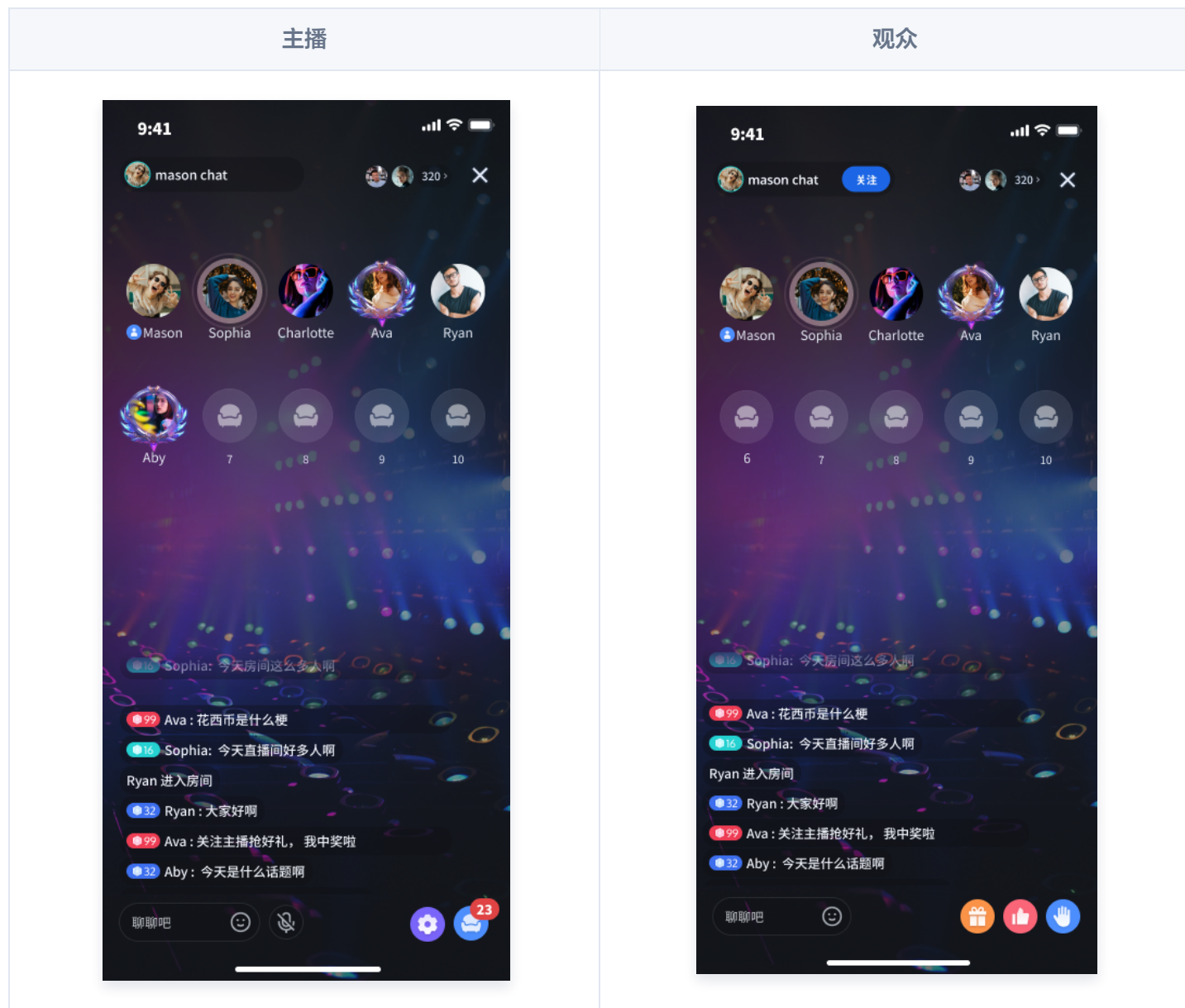
如果您的接入和使用中遇到问题，请参见 [常见问题](#)。



# iOS

最近更新时间：2025-07-14 21:16:52

本文将介绍如何在短时间内完成 TUILiveKit 组件的接入，跟随本文档，您将在一小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的语音聊天室功能。



## 环境准备

- Xcode 15 及以上。
- iOS 13.0 及以上。
- CocoaPods 环境安装，[点击查看](#)。
- 如果您的接入和使用中遇到问题，请参见 [常见问题](#)。

## 步骤1：开通服务

TUILiveKit 目前暂不支持自助开通，如需使用，您可以 [提交申请](#)，申请成功后即可获得14天的体验版，可免费体验 TUILiveKit 的全部功能。

## 步骤2：导入 TUILiveKit 组件

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

1. 请在您的 `Podfile` 文件中添加 `pod 'TUILiveKit'` 依赖，如果您遇到任何问题，请参见 [Example](#) 工程。

### Swift

```
target 'xxxx' do
  ...
  ...
  pod 'TUILiveKit'
end
```

如果您没有 `Podfile` 文件，首先终端 `cd` 到 `xxxx.xcodeproj` 目录，然后通过以下命令创建：

```
pod init
```

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

```
pod install
```

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

```
pod repo update
```

之后执行以下命令，更新组件库的 Pod 版本。

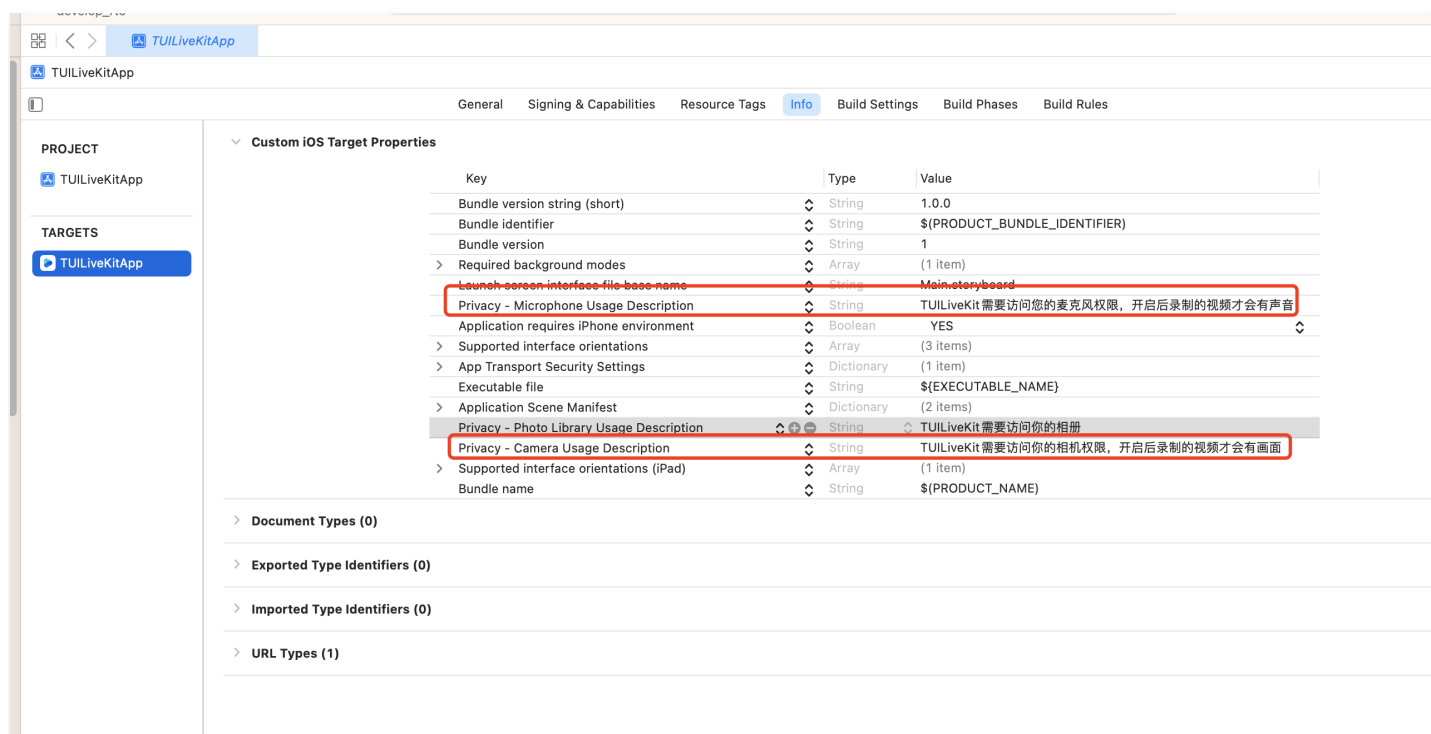
```
pod update
```

3. 可以先编译运行一下，如果遇到问题，请参见 [常见问题](#)。问题如果依然无法解决，可以先去跑一下我们的 [Example](#) 工程。您在接入和使用过程中遇到的任何问题，欢迎给我们 [反馈](#)。

## 步骤3：工程配置

使用音视频功能，需要授权麦克风和摄像头的使用权限。在 App 的 `Info.plist` 中添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

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



## 步骤4：登录

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

### Swift

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

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

```
TUILogin.login(1400000001, // 请替换为步骤一取到的 SDKAppID
               userID: "denny", // 请替换为您的 UserID
               userSig: "xxxxxxxxxxxx") { // 您可以在控制台中计算一个
UserSig 并填在这个位置
    print("login success")
} fail: { (code, message) in
    print("login failed, code: \(code), error: \(message ?? "nil")")
}

return true
}
```

## Objective-C

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

#import <TUICore/TUILogin.h>
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [TUILogin login:1400000001 // 请替换为步骤一取到的 SDKAppID
                  userID:@"denny" // 请替换为您的 UserID
                  userSig:@"xxxxxxxxxxxx" // 您可以在控制台中计算一个 UserSig 并填
在这个位置
                  succ:^(
                        NSLog(@"login success");
                    ) fail:^(int code, NSString * _Nullable msg) {
                        NSLog(@"login failed, code: %d, error: %@", code, msg);
                    }];
    return YES;
}
```

### 参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- **SDKAppID**：在 [步骤一](#) 中的最后一步中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（\_）。
- **UserSig**：使用开通服务（TUILiveKit）获取的 SDKSecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台左侧项目栏中的 [UserSig 工具](#)，创建一个临时可用的 UserSig。

- 更多信息请参见 [如何计算及使用 UserSig](#)。

#### ⚠ 注意:

- 这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：
  - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
  - userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 SDKAppID、userID 以及过期时间等信息加密得来的，而不是直接把 Secretkey 配置成 userSig。
  - userID 被设置成“1”、“123”、“111”等简单字符串，由于 TRTC 不支持同一个 UserID 多端登录，所以在多人协作开发时，形如“1”、“123”、“111”这样的 userID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 userID。
- Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUILiveKit 组件时向您的服务器请求实时计算出的 userSig。

## 步骤5：创建语音聊天室

在上述登录方法调用返回成功后，调用 [VoiceRoomKit](#) 的 [createRoom](#) 方法，创建您的语聊房直播间。

### Swift

```
let params = CreateRoomParams()
params.maxAnchorCount = VoiceRoomDefine.MAX_CONNECTED_VIEWERS_COUNT
params.seatMode = .applyToTake

VoiceRoomKit.createInstance().createRoom(roomId: "your room id", params:
params)
```

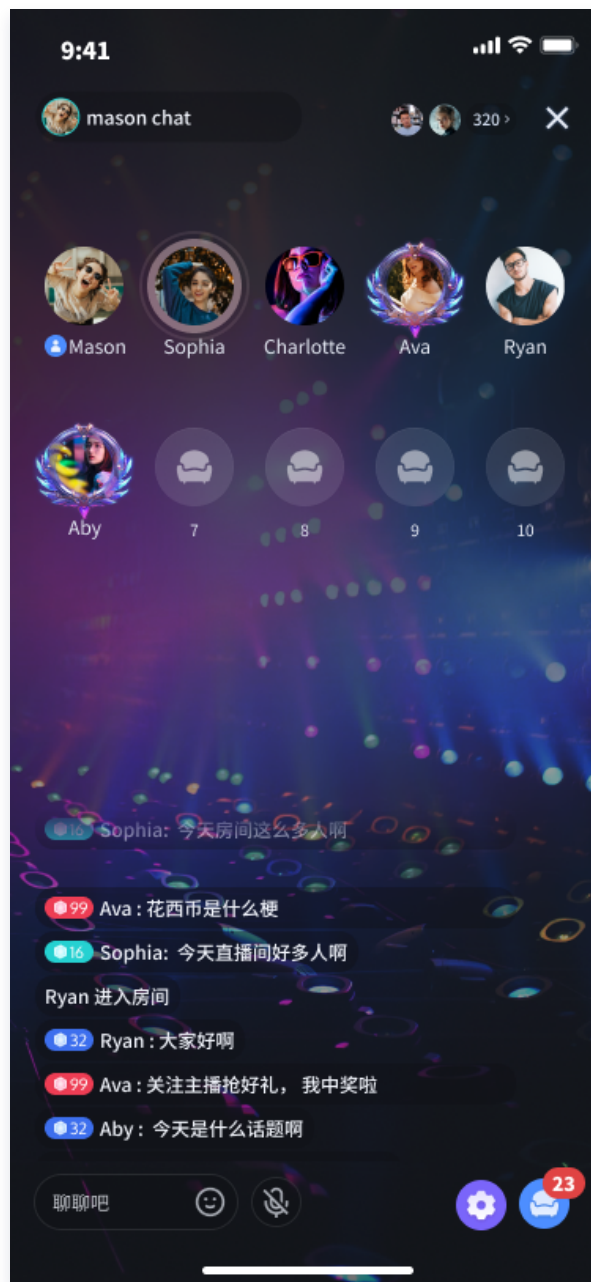
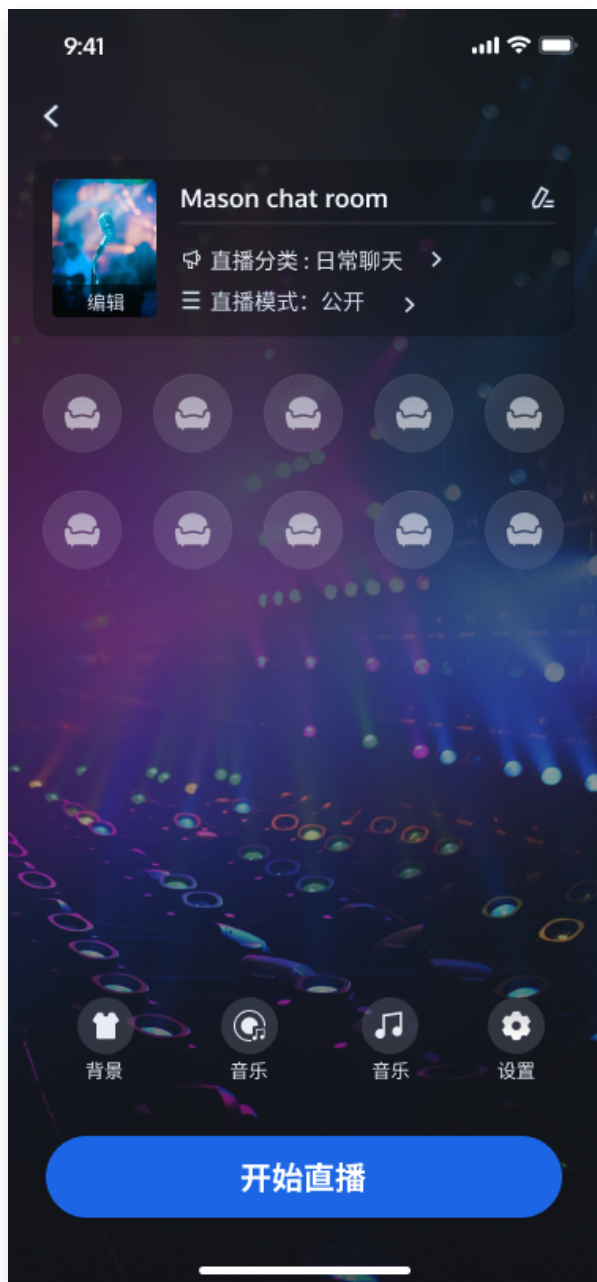
### Objective-C

```
CreateRoomParams *params = [[CreateRoomParams alloc] init];
params.maxAnchorCount = VoiceRoomDefine.MAX_CONNECTED_VIEWERS_COUNT;
params.seatMode = TUISeatModeApplyToTake;

[[VoiceRoomKit createInstance] createRoomWithRoomId:@"your room id"
params:params];
```

语音聊天室预览画面

语音聊天室房间内画面



## 步骤6: 加入语音聊天室

在上述登录方法调用返回成功后，调用 `VoiceRoomKit` 的 `enterRoom` 方法，指定房间 ID，进入语聊房直播间。

### Swift

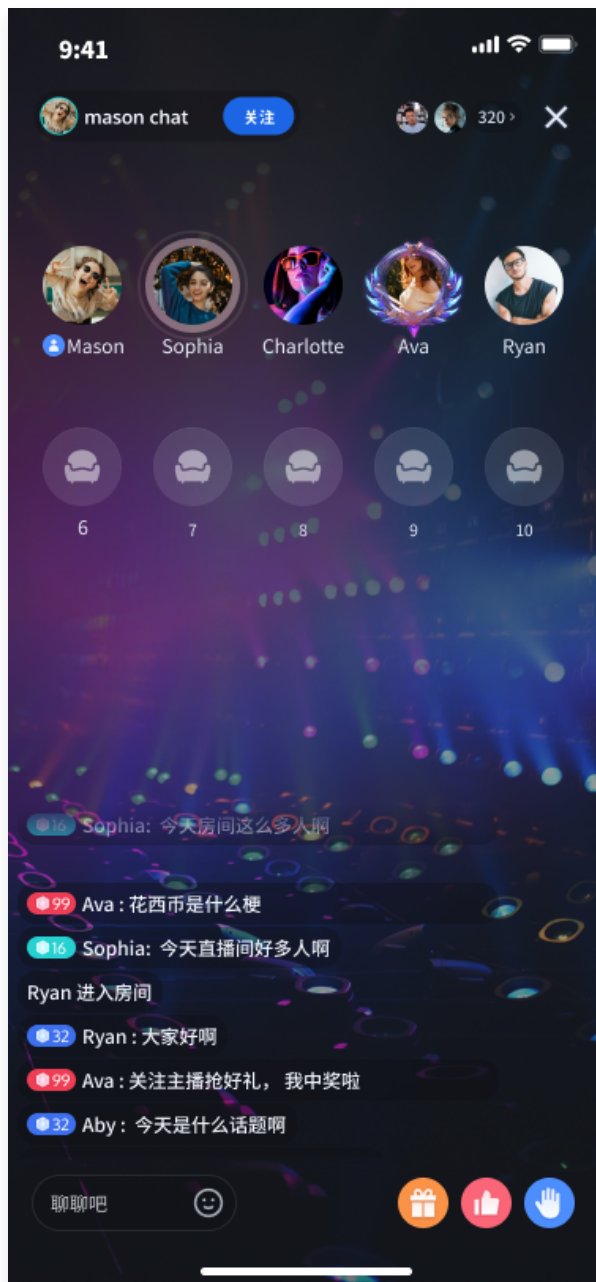
```
VoiceRoomKit.createInstance().enterRoom(roomId: "your room id")
```

### Objective-C

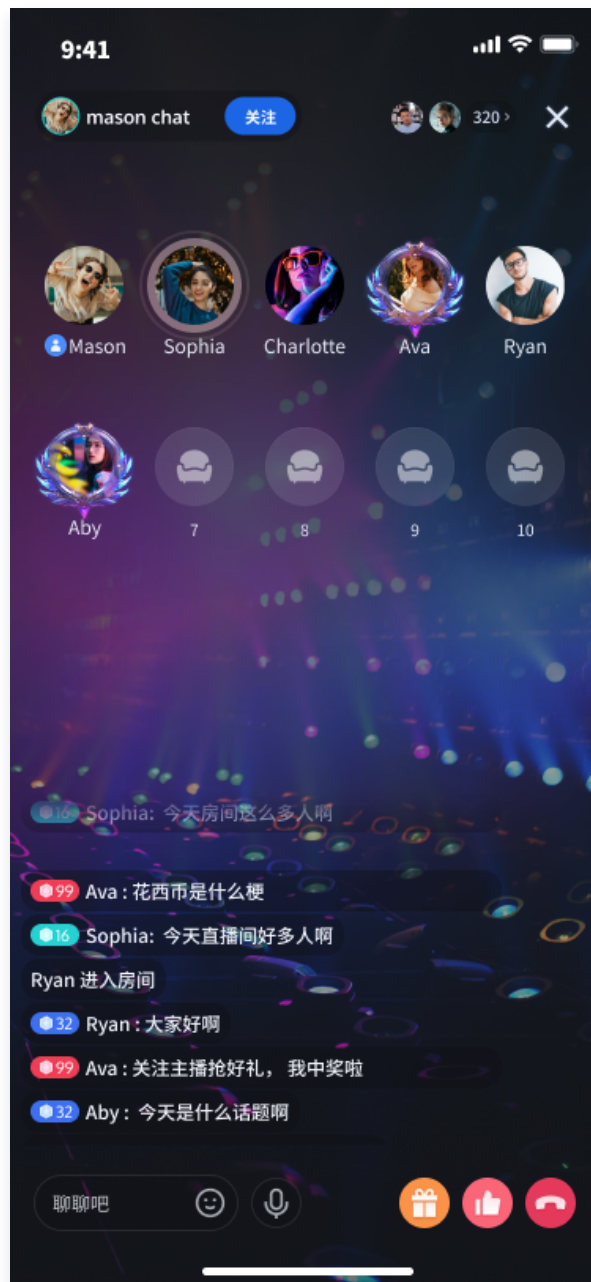


```
[[VoiceRoomKit sharedInstance] enterRoomWithRoomId:@"your room id"];
```

语音聊天室



语音聊天室



## 常见问题

如果您的接入和使用中遇到问题，请参见 [常见问题](#)。

# 界面定制（TUILiveKit）

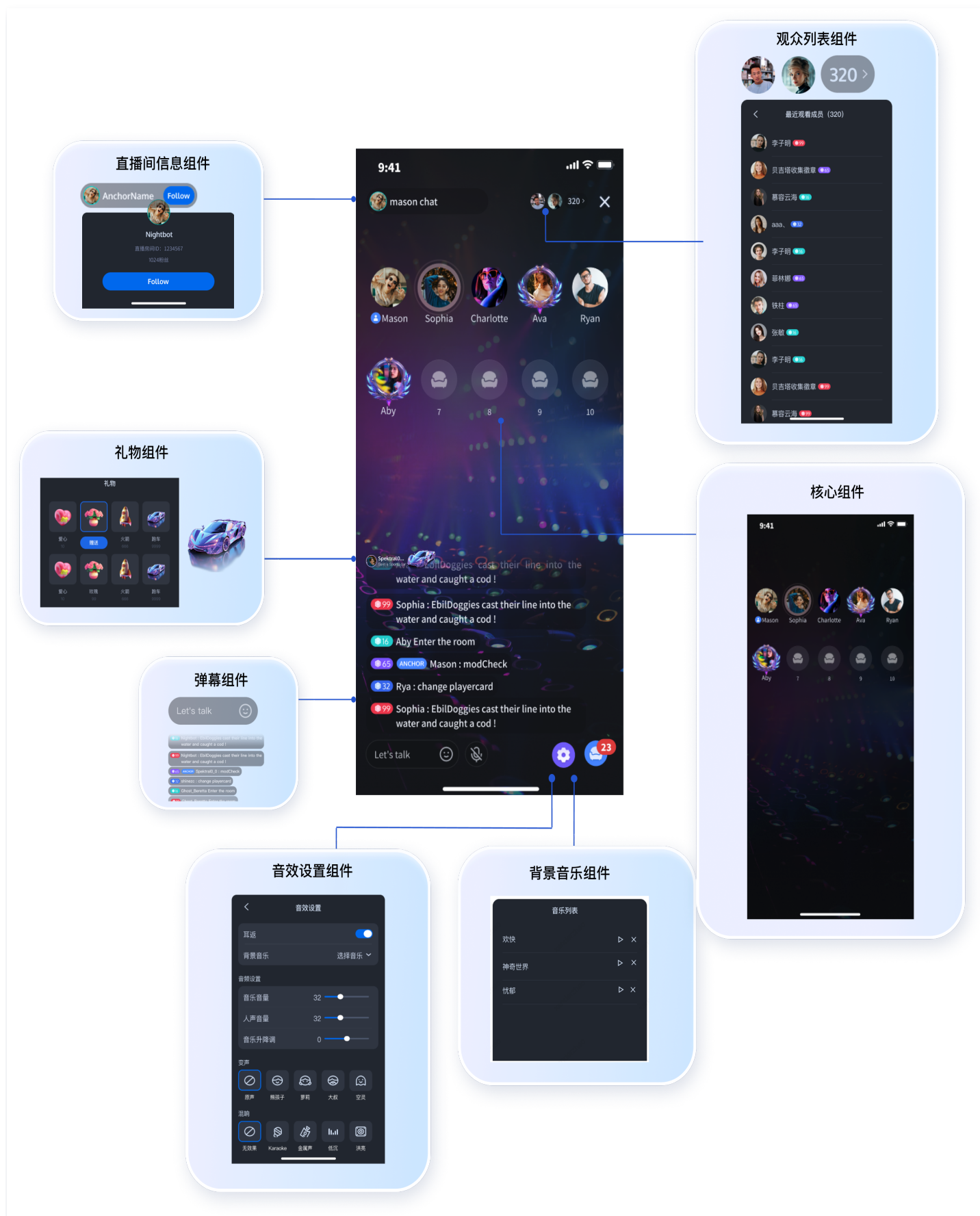
最近更新时间：2025-04-03 17:08:42

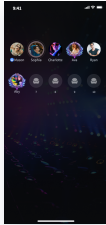
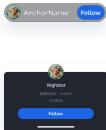
## LiveKit 组件集合介绍

LiveKit 是基于腾讯云 RoomEngine SDK 实现直播场景下一系列的 UI 组件库，它提供了一些通用的 UI 组件，包含核心组件、直播间信息组件、观众列表组件、弹幕组件、礼物组件等。

基于这些 UI 组件您可以像搭积木一样快速搭建起自己的语音聊天室业务逻辑。



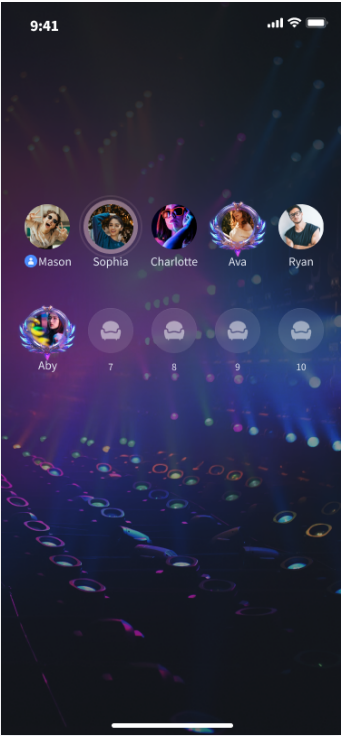
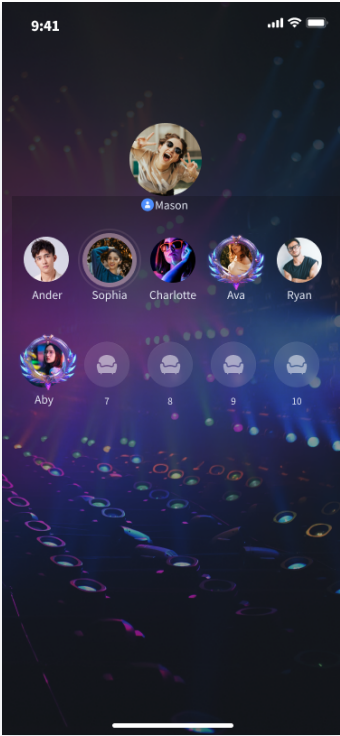
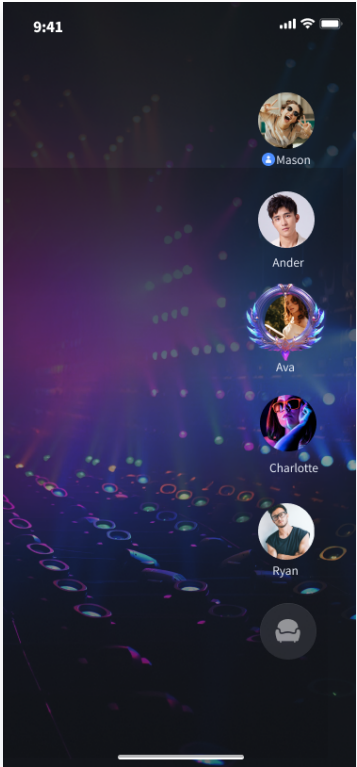


| 组件英文名            | 组件中文名   | 作用                                                                                                                                                              | 缩略图                                                                                                                                                                         |
|------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SeatGridView     | 语聊房核心控件 | 语聊房核心组件提供了开启语音聊天室、关闭语音聊天室，直播间内麦位管理，如申请上麦，邀请上麦，移动麦位，踢人下麦等丰富的 API。                                                                                                |                                                                                          |
| LiveInfoView     | 直播间信息组件 | 直播间信息组件会展示主播信息和直播间信息，并且支持观众最自己喜欢的主播进行关注、取消关注等功能。                                                                                                                |                                                                                          |
| AudienceListView | 观众列表组件  | 观众列表组件会展示直播间最新进房的前100位观众和观众人数，您可以在您的直播界面展示观众列表的简化版视图，当简化版视图被点击时，可以拉起观众列表详情视图。                                                                                   |                                                                                         |
| BarrageView      | 互动弹幕组件  | <ul style="list-style-type: none"> <li>互动弹幕组件支持主播和观众相互发送文本消息或者表情消息。</li> <li>互动弹幕组件提供了 BarrageInputView (弹幕发送 View) 和 BarrageStreamView (弹幕显示 View)。</li> </ul> |                                                                                        |
| GiftView         | 互动礼物组件  | <ul style="list-style-type: none"> <li>互动礼物组件支持直播间的观众向自己欣赏的直播主播送上虚拟礼物，其中礼物支持动效。</li> <li>互动礼物组件提供了 GiftButton (礼物发送按钮) 和 GiftPlayView (礼物播放 View)。</li> </ul>   |   |
| MusicView        | 背景音乐组件  | 音乐列表组件允许您为直播间添加背景音乐。您还可以暂停、重新播放或删除背景音乐。                                                                                                                         |                                                                                        |
| AudioEffectView  | 音效设置组件  | 音效设置组件提供了设置您的直播音量、背景音乐音量、变声等效果。                                                                                                                                 |                                                                                        |

## SeatGridView 组件介绍

SeatGridView 是我们为语音聊天室 UIKit 开发的一个基础控件。这个核心控件提供了丰富的 API，包括开启语音聊天室、关闭语音聊天室，直播间内麦位管理，如申请上麦，邀请上麦，移动麦位，踢人下麦等功能。您只需调用 SeatGridView 组件的几行 API，即可实现以下直播画面。

具体接入 SeatGridView 组件详见 [开播和观看](#) 接入文档。

| 麦位列表-宫格布局                                                                          | 麦位列表-元素布局                                                                          | 麦位列表-纵向布局                                                                            |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|  |  |  |

## 重点组件介绍

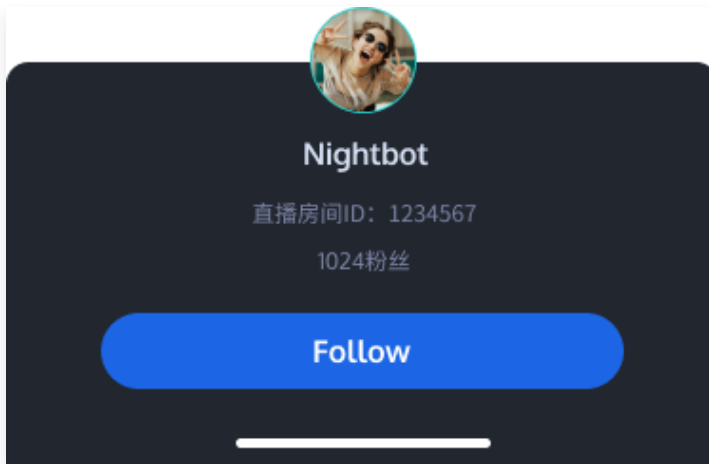
### LiveInfoView 组件

LiveInfoView（直播间信息组件）中展示了主播和直播间的信息，默认情况下展示主播信息视图。当您点击主播信息视图时，将显示直播间信息的详细视图。

- 主播信息视图：



- 直播间信息视图：

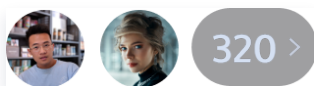


- 直播间信息组件即将上线，敬请期待。

## AudienceListView 组件

AudienceListView（观众列表组件）展示了直播间最新进入的前100位观众及观众总人数。默认情况下展示观众列表视图，点击后将展开显示观众列表的详细视图。

- 观众列表视图



- 观众列表详情

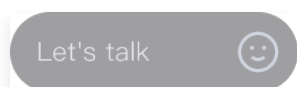


- 观众列表组件即将上线，敬请期待。

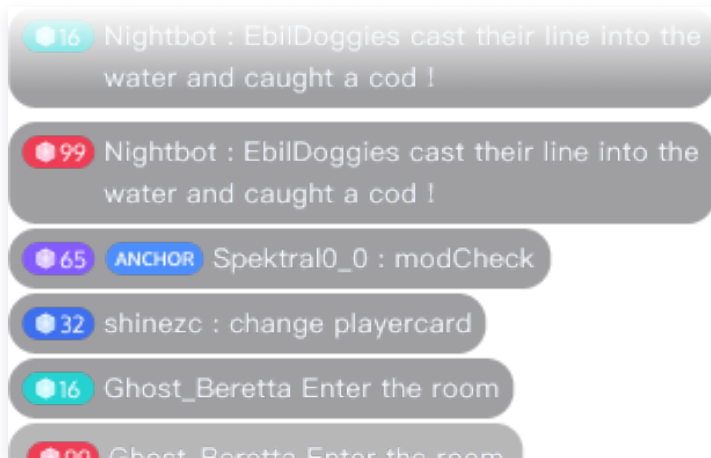
## BarrageView 组件

BarrageView（互动弹幕组件）是一项重要的实时交流工具，支持多种互动方式。用户可以在弹幕中输入表情，增强消息的趣味性，从而使互动体验更加愉快和生动。通过这一功能，观众能够在直播过程中与主播和其他观众进行更为丰富的交流，提升整体的参与感和乐趣。您可以在您想要发送弹幕的布局上添加弹幕发送组件，并在想要显示弹幕的布局上添加弹幕显示组件。

- 弹幕发送组件



- 弹幕显示组件



- 如果您想接入弹幕组件请参见 [互动弹幕](#) 组件接入文档。

## GiftView 组件

GiftView（互动礼物组件）是一个虚拟礼物互动平台，旨在为用户的社交体验增添更多乐趣。通过这个组件，用户可以向自己喜欢的直播主播送出虚拟礼物，以此表达他们的赞赏、喜爱和支持。互动礼物组件支持设置**礼物素材**、**余额显示**、**普通礼物播放与全屏礼物播放**、**充值按钮**等。

- 礼物发送组件



- 礼物显示组件





- 如果您想接入礼物组件请参见 [互动礼物](#) 组件接入文档。

## MusicView 组件

MusicView（音乐列表组件）能够为您的直播间添加背景音乐。您只需在需要背景音乐时展示我们的音乐列表组件，随后便可在我们提供的音乐组件上播放您喜欢的音乐。

- 音乐列表组件



- 背景音乐组件即将上线，敬请期待。

## AudioEffectView 组件



AudioEffectView（音效设置组件）允许您设置直播音量、背景音乐音量和变声效果。只需在需要设置这些效果时展示音效设置组件，便可轻松调整至所需效果。

- 音效设置组件



- 音效设置组件即将上线，敬请期待。

# 开播和收听（TUILiveKit） Android

最近更新时间：2025-04-07 17:43:42

## 适用场景

我们的**主播开播和观众观看功能**主要依赖于**我们语音聊天室的核心控件**(SeatGridView)，该核心控件提供了开启语音聊天室、关闭语音聊天室，直播间内麦位管理，如申请上麦，邀请上麦，移动麦位，踢人下麦等丰富的 API。当您通过 [快速接入](#) 接入语音聊天室 UIKit 后，如果 UI 风格和您的理想中 UI 风格有出入，您可以使用我们的核心控件半小时内快速搭建语音聊天室的主流程。然后在其之上添加您自己的业务 UI 视图。

## 开发环境要求

- Android 5.0（SDK API Level 21）及以上版本。
- Gradle 7.0 及以上的版本。
- Android 5.0 及以上的手机设备。
- 环境配置或编译运行期间，如有问题，请参见 [常见问题](#)。

## 步骤一：开通服务

请参见 [开通服务（TUILiveKit）](#)，领取体验版或者开通付费版。

## 步骤二：工程配置

1. 在 app 目录下找到 `build.gradle.kts` (或 `build.gradle`) 文件，并在其中增加如下代码，加入对 SeatGridView 组件的依赖：

build.gradle.kts

```
api("io.trtc.uikit:live-stream-core:latest.release")
```

build.gradle

```
api 'io.trtc.uikit:live-stream-core:latest.release'
```

2. 由于我们在 SDK 内部使用了 Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在 `proguard-rules.pro` 文件中添加如下代码：

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

```
-keep class com.trtc.uikit.livekit.voiceroomcore.** { *; }  
-keep class com.google.gson.** { *; }
```

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

```
// app/src/main/AndroidManifest.xml  
  
<application  
    ...  
  
    // 添加如下配置覆盖 依赖的 sdk 中的配置  
    android:allowBackup="false"  
    tools:replace="android:allowBackup">
```

## 步骤三：登录

在您的项目中添加如下代码，它的作用是通过调用 TUICore 中的相关接口完成 TUI 组件的登录。这一步骤至关重要，只有在成功登录之后，您才能正常使用 SeatGridView 提供的各项功能。

### Kotlin

```
TUIRoomEngine.login(applicationContext,  
    1400000001,        // 请替换为步骤一取到的 SDKAppID  
    "denny",           // 请替换为您的 UserID  
    "xxxxxxxxxxxx",    // 您可以在控制台中计算一个 UserSig 并填在这个位置  
    object : TUIRoomDefine.ActionCallback() {  
        override fun onSuccess() {  
            Log.i(TAG, "login success")  
        }  
  
        override fun onError(errorCode: Int, errorMessage: String) {  
            Log.e(TAG, "login failed, errorCode: $errorCode  
msg:$errorMessage")  
        }  
    })
```

### Java

```
TUIRoomEngine.login(context,
```

```
1400000001,      // 请替换为步骤一取到的 SDKAppID
"denny",         // 请替换为您的 UserID
"xxxxxxxxxxxx",  // 您可以在控制台中计算一个 UserSig 并填在这个位置
    new TUIRoomDefine.ActionCallback() {
        @Override
        public void onSuccess() {
            Log.i(TAG, "login success");
        }

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

### 参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

| 参数       | 类型     | 说明                                                                                                                                                                                                                  |
|----------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SDKAppID | int    | 在步骤一中的最后一步中您已经获取到，这里不再赘述。                                                                                                                                                                                           |
| UserID   | String | 当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符和下划线。                                                                                                                                                                |
| userSig  | String | 使用 <a href="#">步骤一</a> 的第3步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 <a href="#">辅助工具</a> 生成一个临时可用的 UserSig。更多信息请参见 <a href="#">如何计算及使用 UserSig</a> 。 |

#### ❗ 说明：

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

## 步骤四：使用核心控件实现直播功能

### 创建核心控件

**创建核心控件：**您可以在您 Activity 中通过 java 代码或者 xml 方式加载我们的核心控件，其中代码方式示例如下（XML 方式也类似）：

kotlin

```
val seatGridView = SeatGridView(this)
```

java

```
SeatGridView seatGridView = new SeatGridView(this);
```

## 主播开启直播间并上麦

**主播开启直播间：**开启一个直播间，主播上麦并打开本地麦克风采集。

kotlin

```
val roomInfo = TUIRoomDefine.RoomInfo()
roomInfo.roomId = Config.roomId
roomInfo.seatMode = TUIRoomDefine.SeatMode.APPLY_TO_TAKE
seatGridView.startVoiceRoom(roomInfo, object :
TUIRoomDefine.GetRoomInfoCallback {
    override fun onSuccess(roomInfo: TUIRoomDefine.RoomInfo) {
        val seatIndex = 1
        val timeout = 60
        seatGridView.takeSeat(seatIndex, timeout, null)
        seatGridView.startMicrophone(null)
    }

    override fun onError(error: TUICommonDefine.Error, message: String) {
    }
})
```

java

```
TUIRoomDefine.RoomInfo roomInfo = new TUIRoomDefine.RoomInfo();
roomInfo.roomId = "roomId_123456";
roomInfo.seatMode = TUIRoomDefine.SeatMode.APPLY_TO_TAKE;
seatGridView.startVoiceRoom(roomInfo, new
TUIRoomDefine.GetRoomInfoCallback() {
```

```
@Override
public void onSuccess(TUIRoomDefine.RoomInfo roomInfo) {
    int seatIndex = 1;
    int timeout = 60;
    seatGridView.takeSeat(seatIndex, timeout, null);
    seatGridView.startMicrophone(null);
}

@Override
public void onError(TUICommonDefine.Error error, String message) {
}

});
```

## 观众加入直播间

### kotlin

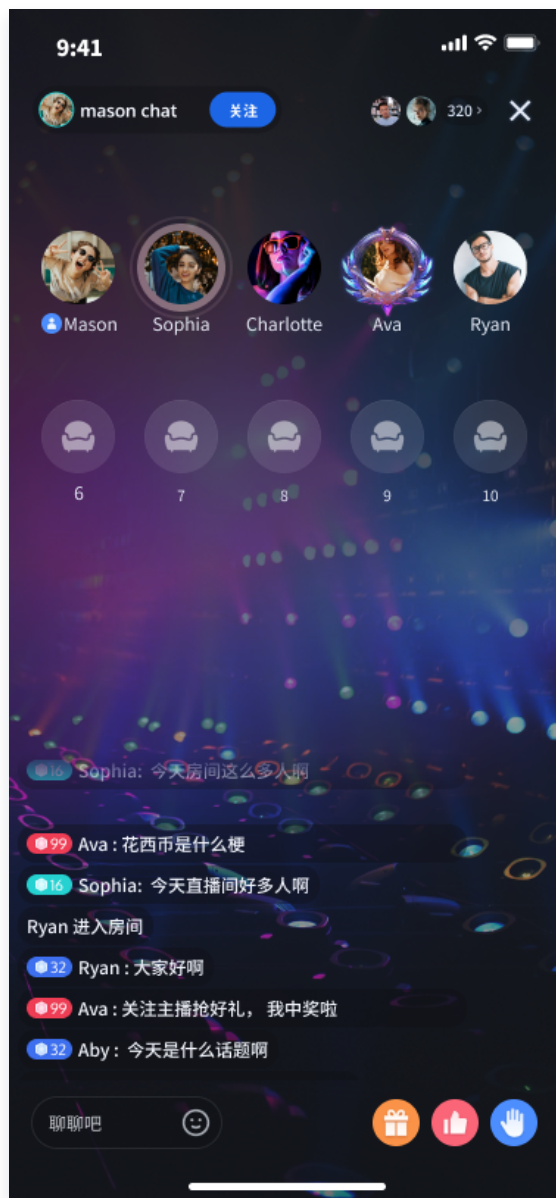
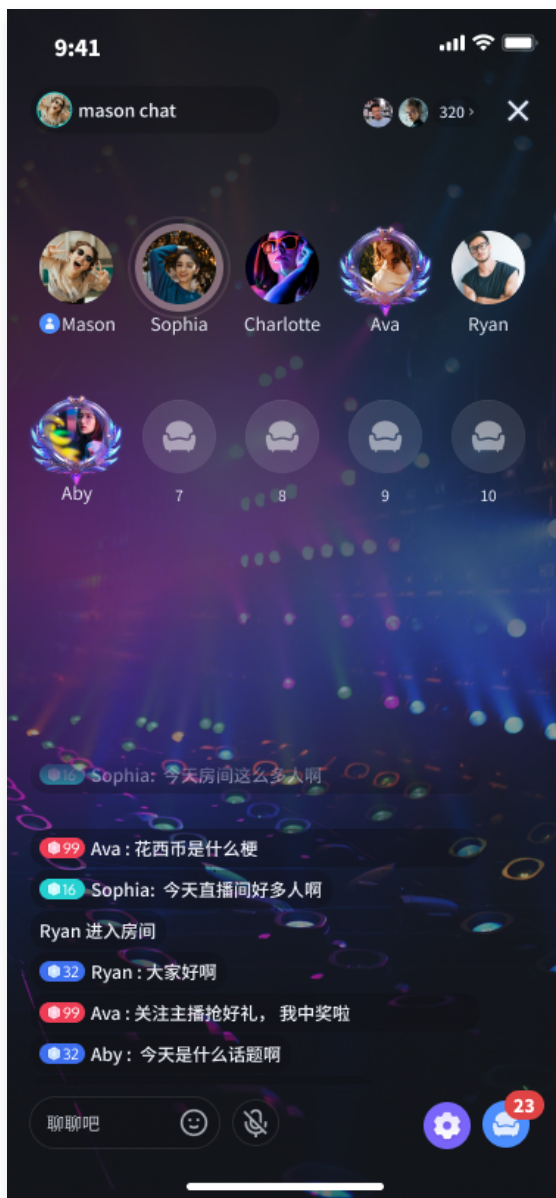
```
seatGridView.joinVoiceRoom("roomId_123456", null)
```

### java

```
seatGridView.joinVoiceRoom("roomId_123456", null);
```

主播开启直播间开始直播

观众加入直播间观看直播



## 麦位管理

您如果需要实现麦位管理功能，可参见 [麦位管理](#) 文档。

## 设置麦位列表布局排列

您可以通过以下方式快速设置您的麦位列表布局。

kotlin

```
// 设置宫格布局
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.GRID, null)
```



```
// 设置元素布局
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FOCUS, null)

// 设置纵向布局
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.VERTICAL, null)

// 设置自定义布局
val layoutConfig = VoiceRoomDefine.SeatViewLayoutConfig().apply {
    rowConfigs = ArrayList()
    rowSpacing = dp2px(10); //每行的间距
}

// 第一行配置
val rowConfig1 = VoiceRoomDefine.SeatViewLayoutRowConfig().apply {
    count = 3 //第一行显示的数量
    seatSize = VoiceRoomDefine.Size(dp2px(50), dp2px(50)) //第一行显示的每个麦位视图大小
    seatSpacing = dp2px(10) //第一行每个麦位的水平间距
    alignment = VoiceRoomDefine.SeatViewLayoutRowAlignment.CENTER //第一行麦位的对齐方式
}
layoutConfig.rowConfigs.add(rowConfig1)

// 第二行配置
val rowConfig2 = VoiceRoomDefine.SeatViewLayoutRowConfig().apply {
    count = 3 //第二行显示的数量
    seatSize = VoiceRoomDefine.Size(dp2px(50), dp2px(50)) //第二行显示的每个麦位大小
    seatSpacing = dp2px(10) //第二行每个麦位的水平间距
    alignment = VoiceRoomDefine.SeatViewLayoutRowAlignment.SPACE_AROUND
} //第二行麦位的对齐方式
layoutConfig.rowConfigs.add(rowConfig2)
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FREE, layoutConfig)
```

## java

```
// 设置宫格布局
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.GRID, null)

// 设置元素布局
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FOCUS, null)

// 设置纵向布局
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.VERTICAL, null)
```

```
// 设置自由布局
VoiceRoomDefine.SeatViewLayoutConfig layoutConfig = new
VoiceRoomDefine.SeatViewLayoutConfig();
layoutConfig.rowConfigs = new ArrayList<>();
layoutConfig.rowSpacing = dp2px(10); //每行的间距
//第一行配置
VoiceRoomDefine.SeatViewLayoutRowConfig rowConfig1 = new
VoiceRoomDefine.SeatViewLayoutRowConfig();
rowConfig1.count = 3; //第一行显示的数量
rowConfig1.seatSize = new VoiceRoomDefine.Size(dp2px(50), dp2px(50)); //第
一行显示的每个麦位视图大小
rowConfig1.seatSpacing = dp2px(10); //第一行每个麦位的水平间距
rowConfig1.alignment = VoiceRoomDefine.SeatViewLayoutRowAlignment.CENTER;
//第一行麦位的对齐方式
layoutConfig.rowConfigs.add(rowConfig1);
//第二行配置
VoiceRoomDefine.SeatViewLayoutRowConfig rowConfig2 = new
VoiceRoomDefine.SeatViewLayoutRowConfig();
rowConfig2.count = 3; //第二行显示的数量
rowConfig2.seatSize = new VoiceRoomDefine.Size(dp2px(50), dp2px(50)); //第
二行显示的每个麦位视图大小
rowConfig1.seatSpacing = dp2px(10); //第二行每个麦位的水平间距
rowConfig2.alignment =
VoiceRoomDefine.SeatViewLayoutRowAlignment.SPACE_AROUND; //第二行麦位的对齐方
式
layoutConfig.rowConfigs.add(rowConfig2);
seatGirdView.setLayoutMode(VoiceRoomDefine.LayoutMode.FREE, layoutConfig);
```

**❗ 说明：**

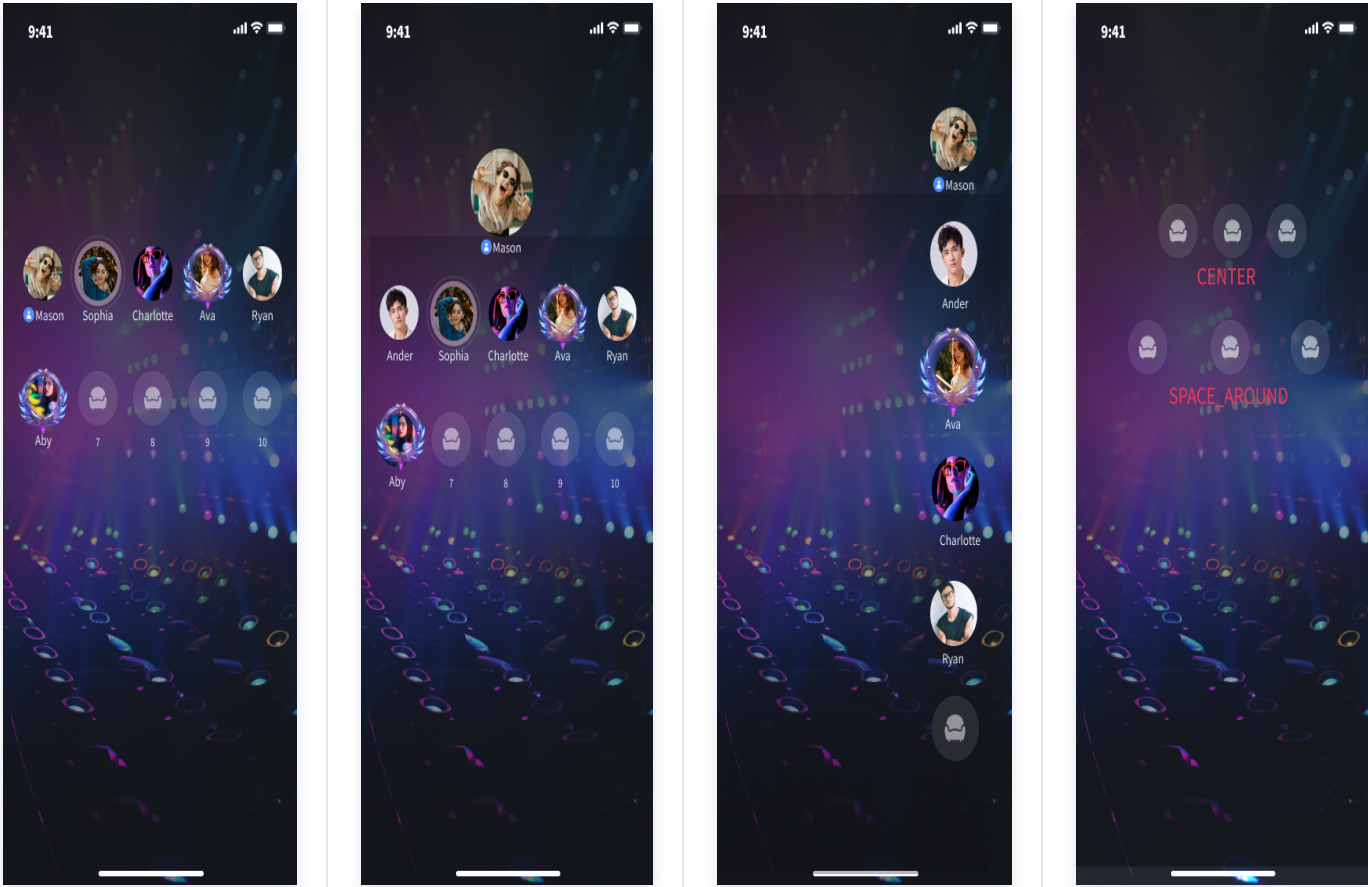
自定义布局的参数设置可查看 [SeatViewLayoutRowConfig](#) 中参数说明，其中对齐方式 alignment 可参见 [SeatViewLayoutRowAlignment](#) 中描述对齐方式效果可参见 [自定义布局对齐方式示意图](#)。

宫格布局

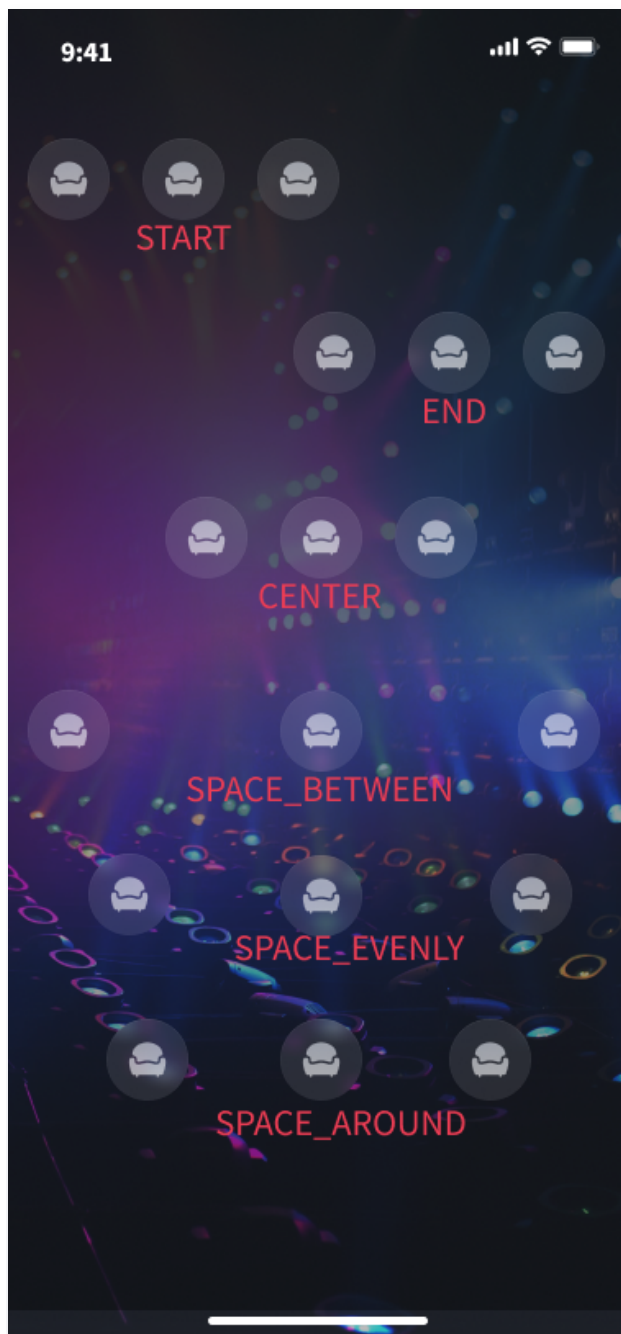
元素布局

纵向布局

自定义布局



自定义布局对齐方式示意图：



## 自定义麦位视图

如果您认为我们默认的UI不满足您的需求，您想自定义自己的麦位UI，您可以通过以下方式快速设置您的麦位布局，完全自定义自己的麦位视图UI。

kotlin

```
val adapter = object : VoiceRoomDefine.SeatViewAdapter {  
    override fun createSeatView(seatGridView: SeatGridView, seatInfo:  
TUIRoomDefine.SeatInfo): View {  
        return TestSeatInfoView(context, seatGridView, seatInfo)  
    }  
}
```

```
}

    override fun updateSeatView(seatGridView: SeatGridView, seatInfo:
TUIRoomDefine.SeatInfo, seatView: View) {
        (seatView as TestSeatInfoView).updateSeatView(seatGridView,
seatInfo)
    }

    override fun updateUserVolume(seatGridView: SeatGridView, volume: Int,
customSeatView: View) {
        (customSeatView as
TestSeatInfoView).updateUserVolume(seatGridView, volume)
    }
}

seatGirdView.setSeatViewAdapter(adapter)

class TestSeatInfoView constructor(context: Context, seatGirdView:
SeatGridView, seatInfo: TUIRoomDefine.SeatInfo) : FrameLayout(context) {
    init {
        initView() //初始化view
    }

    fun updateSeatView(seatGirdView: SeatGridView, seatInfo:
TUIRoomDefine.SeatInfo) {
        updateView(seatInfo) //更新自定义麦位视图UI
    }

    fun updateUserVolume(seatGirdView: SeatGridView, volume: Int) {
        updateUserVolume(volume) //更新音量变化UI
    }
}
```

## java

```
VoiceRoomDefine.SeatViewAdapter adapter = new
VoiceRoomDefine.SeatViewAdapter() {
    @Override
    public View createSeatView(SeatGridView seatGridView,
TUIRoomDefine.SeatInfo seatInfo) {
        return new TestSeatInfoView(getApplicationContext(), seatGridView,
seatInfo);
    }
}
```

```
@Override
public void updateSeatView(SeatGridView seatGridView,
    TUIRoomDefine.SeatInfo seatInfo,
    View customSeatView) {
    ((TestSeatInfoView) customSeatView).updateSeatView(seatGridView,
seatInfo);
}

@Override
public void updateUserVolume(SeatGridView seatGridView, int volume,
    View customSeatView) {
    ((TestSeatInfoView) customSeatView).updateUserVolume(seatGridView,
volume);
}
};
seatGirdView.setSeatViewAdapter(adapter);

public class TestSeatInfoView extends FrameLayout {

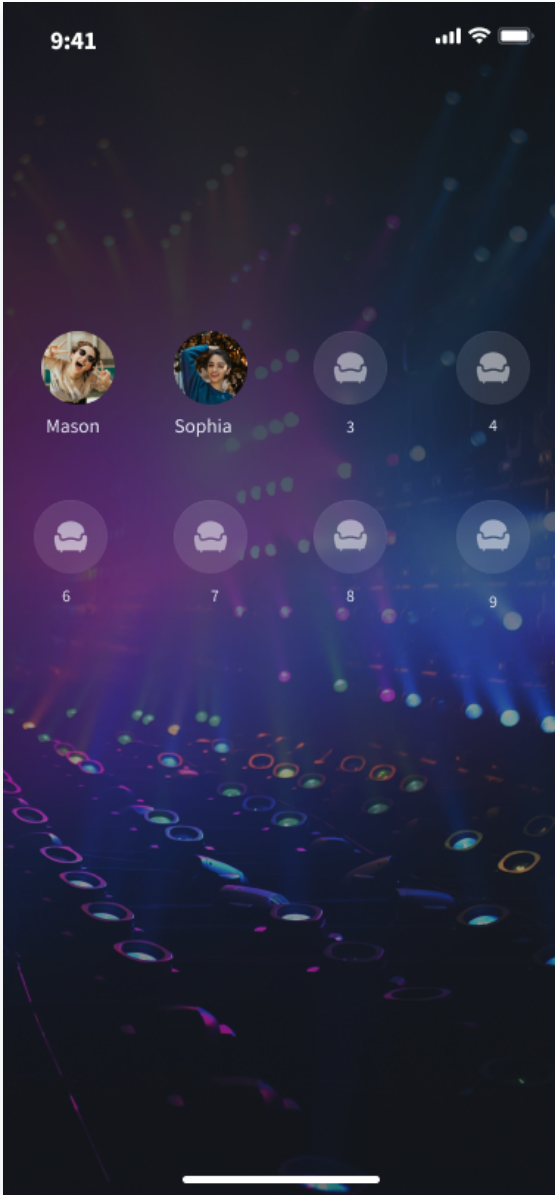
    public TestSeatInfoView(@NonNull Context context, SeatGridView
seatGirdView, TUIRoomDefine.SeatInfo seatInfo) {
        super(context);
        initView();
    }

    public void updateSeatView(SeatGridView seatGirdView,
    TUIRoomDefine.SeatInfo seatInfo) {
        updateView(seatInfo);
    }

    public void updateUserVolume(SeatGridView seatGirdView, int volume) {
        updateUserVolume(volume);
    }
}
```

默认麦位视图

自定义麦位视图示例





# iOS

最近更新时间：2025-05-29 16:17:21

## 适用场景

我们的**主播开播和观众观看功能**主要依赖于**我们核心控件(LiveCoreView)**，该核心控件提供了开启语音聊天室、关闭语音聊天室，直播间内麦位管理，如申请上麦，邀请上麦，移动麦位，踢人下麦等丰富的 API。

当您通过 **快速接入** 接入语音聊天室 UIKit 后，如果 UI 风格和您的理想中 UI 风格有出入，您可以使用我们的核心控件半小时内快速搭建语音聊天室的主流程。然后在其之上添加您自己的业务 UI 视图。

## 环境准备

- Xcode 15 及以上。
- iOS 13.0 及以上。
- CocoaPods 环境安装，[点击查看](#)。
- 如果您的接入和使用中遇到问题，请参见 [常见问题](#)。

## 步骤一：开通服务

请参见 [开通服务（TUILiveKit）](#)，领取体验版或者开通付费版。

## 步骤二：导入 LiveStreamCore 组件

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

1. 请在您的 `Podfile` 文件中添加 `pod 'LiveStreamCore'` 依赖。

### Swift

```
target 'xxxx' do
  ...
  ...
  pod 'LiveStreamCore'
end
```

如果您没有 `Podfile` 文件，首先终端 `cd` 到 `xxxx.xcodeproj` 目录，然后通过以下命令创建：

```
pod init
```

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

```
pod install
```

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

```
pod repo update
```

之后执行以下命令，更新组件库的 Pod 版本。

```
pod update
```

3. 可以先编译运行一下，如果遇到问题，请参见 [常见问题](#)。问题如果依然无法解决，可以先去跑一下我们的 [Example](#) 工程。您在接入和使用过程中遇到的任何问题，欢迎给我们 [反馈](#)。

## 步骤三：登录

在您的项目中添加如下代码，它的作用是通过调用 RTCRoomEngine 中的登录相关接口完成 TUI 组件的登录。这一步骤至关重要，只有在成功登录之后，您才能正常使用 LiveCoreView 提供的各项功能。

swift

```
//  
// AppDelegate.swift  
//  
  
import RTCRoomEngine  
  
func application(_ application: UIApplication,  
didFinishLaunchingWithOptions launchOptions:  
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {  
  
    TUIRoomEngine.login(sdkAppId: 1400000001, // 请替换为步骤一取到的  
SDKAppID  
                        userId: "denny", // 请替换为您的 UserID  
                        userSig: "xxxxxxxxxxxx") { // 您可以在控制台中计算一  
个 UserSig 并填在这个位置  
        print("login success")  
    } onError: { code, message in  
        print("login failed, code: \(code), error: \(message ?? "nil")")  
    }  
}
```

```
return true
}
```

### 参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

| 参数       | 类型     | 说明                                                                                                                                                                                                                  |
|----------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SDKAppID | int    | 在步骤一中的最后一步中您已经获取到，这里不再赘述。                                                                                                                                                                                           |
| UserID   | String | 当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符和下划线。                                                                                                                                                                |
| userSig  | String | 使用 <a href="#">步骤一</a> 的第3步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 <a href="#">辅助工具</a> 生成一个临时可用的 UserSig。更多信息请参见 <a href="#">如何计算及使用 UserSig</a> 。 |

#### ❗ 说明：

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

## 步骤四：使用核心控件实现直播功能

### 创建核心控件并开启预览

创建核心控件：

```
swift
```

```
import LiveStreamCore

let seatGridView = SeatGridView()
```

### 主播开启直播间和观众加入直播间

**主播开启直播间：**开启一个直播间，并将本地麦克风采集的数据推流到直播间。

## swift

```
import LiveStreamCore
import RTCRoomEngine

let roomInfo = TUIRoomInfo()
roomInfo.roomId = "123456"
roomInfo.seatMode = .applyToTake

seatGridView.startVoiceRoom(roomInfo: roomInfo) { roomInfo in
} onError: { code, message in
}

seatGridView.startMicrophone() {
} onError: { code,message in
}
```

观众加入直播间。

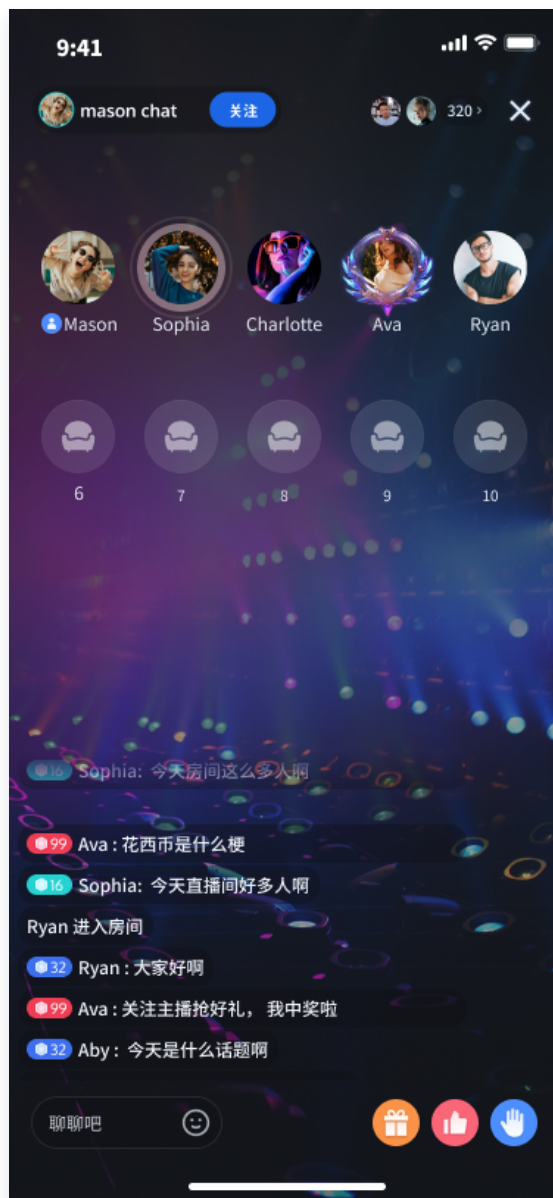
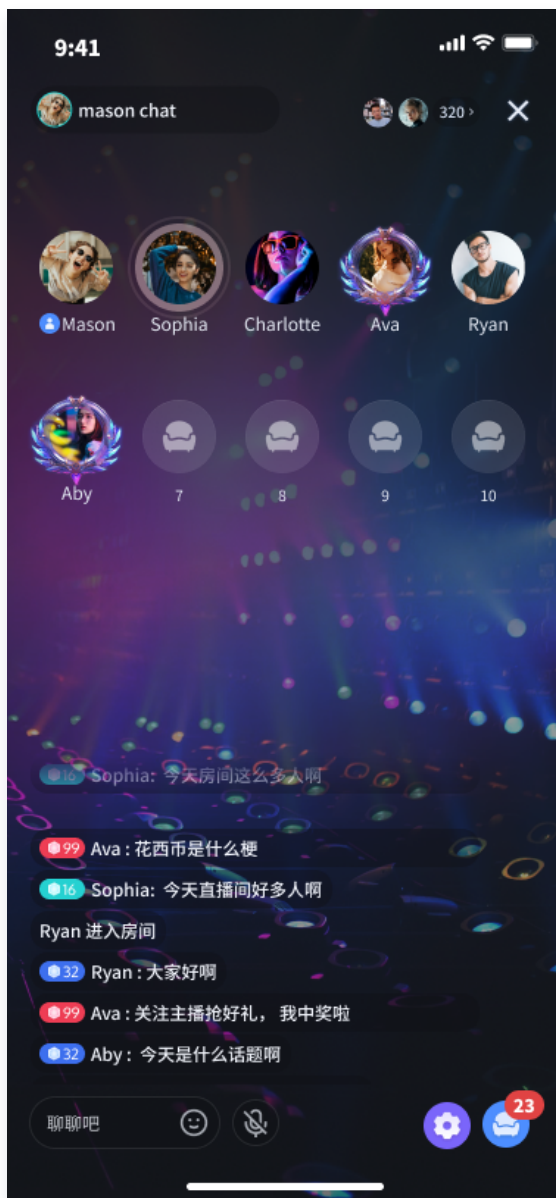
## swift

```
import LiveStreamCore

seatGridView.joinVoiceRoom(roomId: "roomId_123456") { roomInfo in
} onError: { code, message in
}
```

主播开启直播间开始直播

观众加入直播间观看直播



## 观众连麦

您如果需要实现观众连麦功能，可参见 [观众连麦](#) 文档。

## 设置麦位列表排列布局

您可以通过以下方式快速设置您的麦位列表布局。

swift

```
import LiveStreamCore
```

```
// 设置宫格布局
```

```
seatGridView.setLayoutMode(layoutMode: .grid)

// 设置元素布局
seatGridView.setLayoutMode(layoutMode: .focus)

// 设置纵向布局
seatGridView.setLayoutMode(layoutMode: .vertical)

// 设置自定义布局
// 第一行配置
let rowConfig1 = SGSeatViewLayoutRowConfig(count: 3, //第一行显示的数量
                                           seatSpacing: 10, //第一行每个麦位
                                           seatSize: CGSize(width: 50,
                                                             height: 50), //第一行显示的每个麦位视图大小
                                           alignment: .center) //第一行麦位
                                           的对齐方式

// 第二行配置
let rowConfig2 = SGSeatViewLayoutRowConfig(count: 3, //第二行显示的数量
                                           seatSpacing: 10, //第二行每个麦位
                                           seatSize: CGSize(width: 50,
                                                             height: 50), //第二行显示的每个麦位视图大小
                                           alignment: .spaceAround) //第二
                                           行麦位的对齐方式

let layoutConfig = SGSeatViewLayoutConfig(rowConfigs: [rowConfig1,
                                                         rowConfig2],
                                           rowSpacing: 10)

seatGridView.setLayoutMode(layoutMode: .free, layoutConfig: layoutConfig)
```

**❗ 说明:**

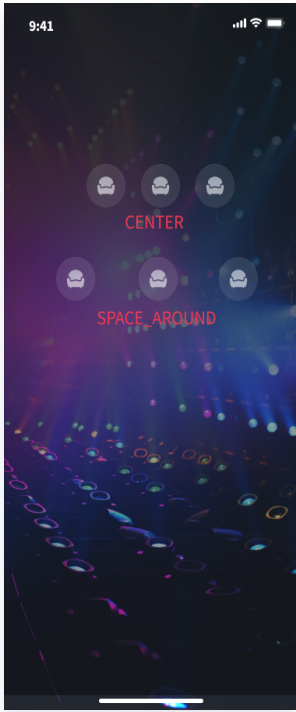
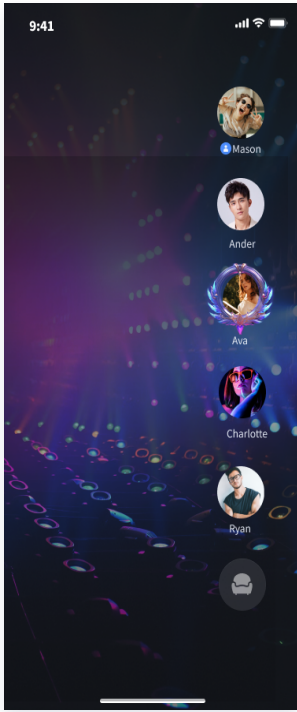
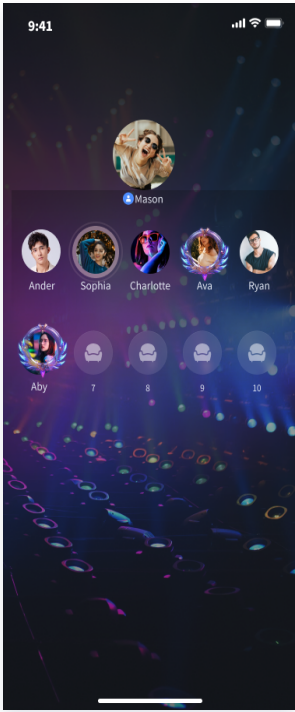
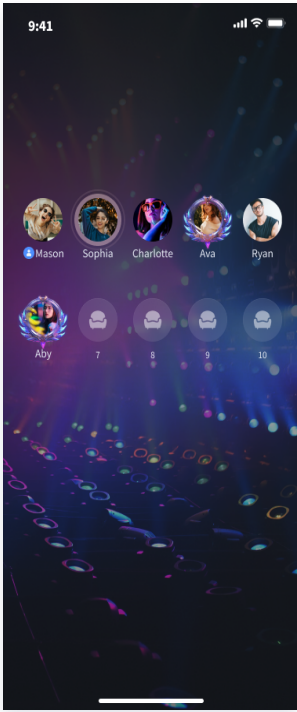
自定义布局的参数设置可查看 [SGSeatViewLayoutRowConfig](#) 中参数说明，其中对齐方式 alignment 可参见 [SGSeatViewLayoutRowAlignment](#) 中描述对齐方式效果可参见 [示意图](#)。

宫格布局

元素布局

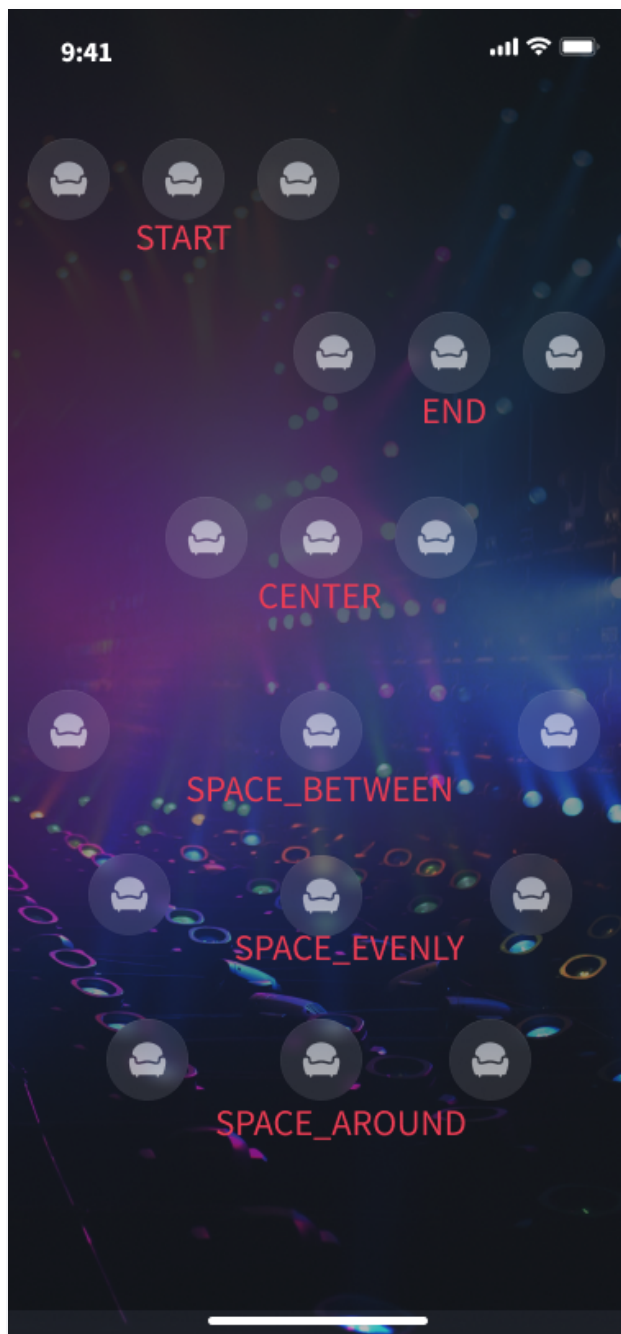
纵向布局

自定义布局



自定义布局对齐方式示意图





## 自定义麦位视图

如果您认为我们默认的UI不满足您的需求，您想自定义自己的麦位UI，您可以通过以下方式快速设置您的麦位布局。

swift

```
import LiveStreamCore

class TestSeatViewDelegate: SGSeatViewDelegate {
    func seatGridView(_ view: SeatGridView, createSeatView seatInfo:
    TUISeatInfo) -> UIView? {
        return TestSeatInfoView(seatGridView: view, seatInfo: seatInfo)
    }
}
```

```
}

func seatGridView(_ view: SeatGridView, updateSeatView seatInfo:
TUISeatInfo, seatView: UIView) {
    if let seatView = seatView as? TestSeatInfoView {
        seatView.updateSeatView(seatGridView: view, seatInfo: seatInfo)
    }
}

func seatGridView(_ view: SeatGridView, updateUserVolume volume: Int,
seatView: UIView) {
    if let seatView = seatView as? TestSeatInfoView {
        seatView.updateUserVolume(seatGridView: view, volume: volume)
    }
}

seatGridView.setSeatViewDelegate(TestSeatViewDelegate())

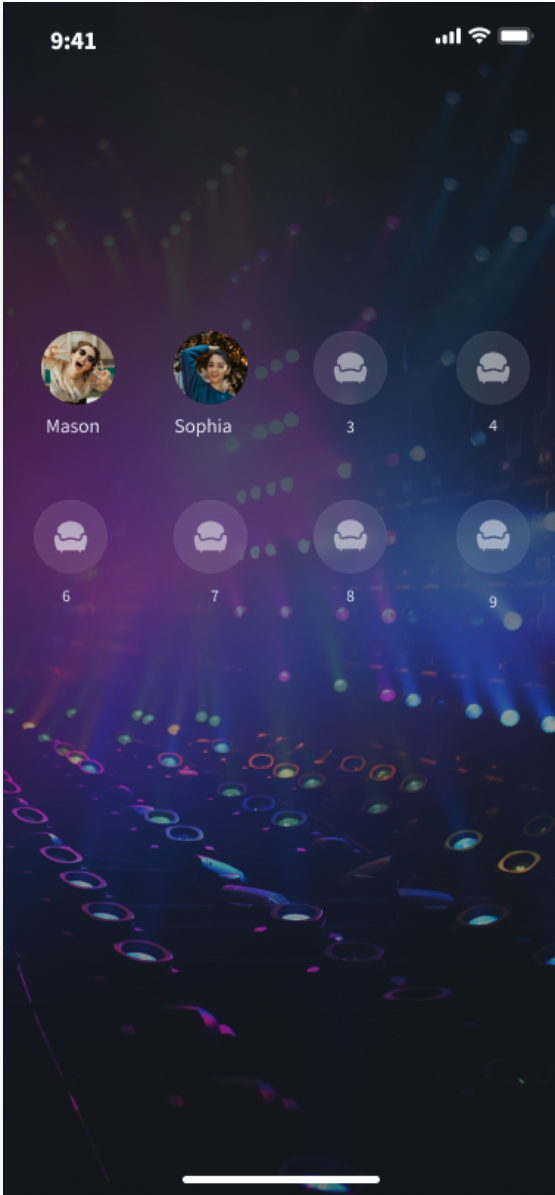
class TestSeatInfoView: UIView {
    init(seatGridView: SeatGridView, seatInfo: TUISeatInfo) {
        super.init(frame: .zero)
        initView() //初始化view
    }

    func updateSeatView(seatGridView: SeatGridView, seatInfo: TUISeatInfo)
{
        updateView(seatInfo) //更新自定义麦位视图UI
    }

    func updateUserVolume(seatGridView: SeatGridView, volume: Int) {
        updateUserVolume(volume) //更新音量变化UI
    }
}
```

自定义麦位布局前

自定义麦位布局后



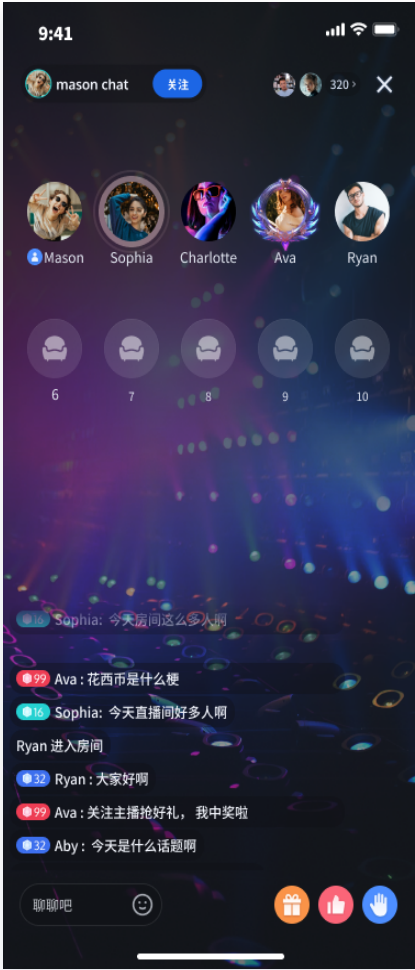
# 麦位管理（TUILiveKit）

最近更新时间：2025-04-17 17:04:12

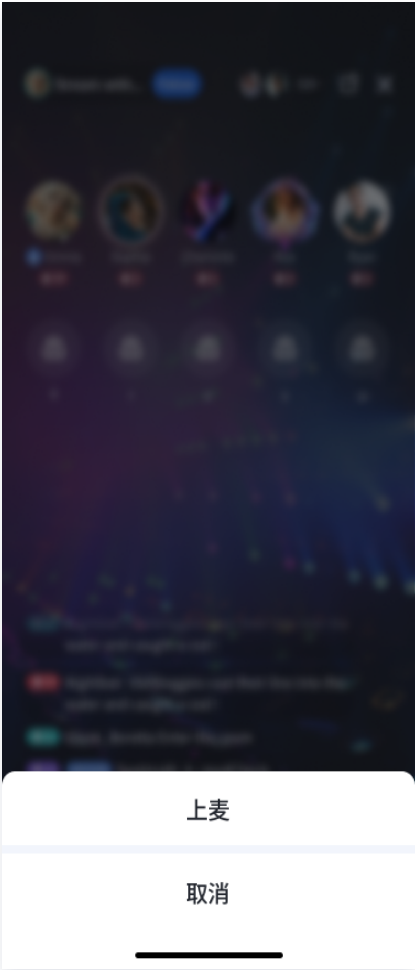
## 功能介绍

麦位管理是一种实时互动交流方式，主播可以麦位上的观众进行实时互动，无论是解答问题、分享经验还是进行娱乐互动，都能极大地提升观众的**参与感和满意度**。这种直接的互动和交流为商业化运营提供了更加便捷和高效的渠道，使用此功能，可以实现申请上麦，邀请上麦，移动麦位，踢人下麦，锁定麦位能功能，极大的丰富了语音聊天室的可玩性。

麦位列表



观众申请上麦



主播处理上麦



## 接入流程

### 观众申请上麦流程

TUILiveKit 观众申请上麦功能主要是通过 `SeatGridView(Android & iOS)` / `SeatGridController(Flutter)` 实现，您可以调用如下 API 函数，实现观众申请上麦功能。以观众 B 申请上麦为例实现如下。

## 观众发送申请上麦请求

观众 B 给主播 A 发送申请上麦请求，主播 A 会在 `onSeatRequestReceived` 回调中收到观众 B 的上麦请求。

### Android

#### Kotlin

```
val seatIndex = 1;
val timeout = 60;
seatGridView.takeSeat(seatIndex, timeout, object :
VoiceRoomDefine.RequestCallback {
    override fun onAccepted(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "申请上麦被同意")
    }

    override fun onRejected(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "申请上麦被拒绝")
    }

    override fun onCancelled(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "申请上麦被取消")
    }

    override fun onTimeout(userInfo: TUIRoomDefine.UserInfo) {
        Log.i(TAG, "申请上麦超时")
    }

    override fun onError(userInfo: TUIRoomDefine.UserInfo, error:
TUICommonDefine.Error, message: String) {
        Log.i(TAG, "申请上麦错误")
    }
})
```

#### Java

```
int seatIndex = 1;
```

```
int timeout = 60;
seatGridView.takeSeat(seatIndex, timeout, new
VoiceRoomDefine.RequestCallback() {
    @Override
    public void onAccepted(TUIRoomDefine.UserInfo userInfo) {
        Log.i(TAG, "申请上麦被同意");
    }

    @Override
    public void onRejected(TUIRoomDefine.UserInfo userInfo) {
        Log.i(TAG, "申请上麦被拒绝");
    }

    @Override
    public void onCancelled(TUIRoomDefine.UserInfo userInfo) {
        Log.i(TAG, "申请上麦被取消");
    }

    @Override
    public void onTimeout(TUIRoomDefine.UserInfo userInfo) {
        Log.i(TAG, "申请上麦超时");
    }

    @Override
    public void onError(TUIRoomDefine.UserInfo userInfo,
TUICommonDefine.Error error, String message) {
        Log.i(TAG, "申请上麦错误");
    }
});
```

## iOS

### swift

```
import RTCRoomEngine
import LiveStreamCore

let seatIndex = 1
let timeout = 60
seatGridView.takeSeat(index: index, timeout: timeout) { userInfo in
```

```
print("申请上麦被同意")
} onRejected: { userInfo in
  print("申请上麦被拒绝")
} onCancelled: { userInfo in
  print("申请上麦被取消")
} onTimeout: { userInfo in
  print("申请上麦超时")
} onError: { userInfo, code, message in
  print("申请上麦错误")
}
```

## Flutter

### Dart

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

const int seatIndex = 1;
const int timeout = 60;
final controller = SeatGridController();
final result = await controller.takeSeat(seatIndex, timeout);
if (result.code == TUIError.success) {
  switch (result.type) {
    case RequestResultType.onAccepted:
      debugPrint('申请上麦被同意');
      break;
    case RequestResultType.onRejected:
      debugPrint('申请上麦被拒绝');
      break;
    case RequestResultType.onCancelled:
      debugPrint('申请上麦被取消');
      break;
    case RequestResultType.onTimeout:
      debugPrint('申请上麦超时');
      break;
    default:
      break;
  }
} else {
```

```
debugPrint('申请上麦错误');  
}
```

TUILiveKit 观众申请上麦功能 主要是通过 `SeatGridView(Android & iOS)` / `SeatGridController(Flutter)` 实现，您可以调用如下 API 函数，实现观众申请上麦功能。以观众 B 申请上麦为例实现如下。

**⚠ 注意：**

只有当房间模式是 `APPLY_TO_TAKE` (申请上麦)，房主才会收到上麦请求，`FREE_TO_TAKE` (自由上麦) 模式下，`takeSeat` 会直接上麦成功。

## 主播端收到申请上麦请求

主播 A 会在 `onSeatRequestReceived` 回调方法中收到 观众B 的上麦请求。

### Android

#### Kotlin

```
override fun onSeatRequestReceived(type: VoiceRoomDefine.RequestType,  
    userInfo: TUIRoomDefine.UserInfo) {  
    if (type == VoiceRoomDefine.RequestType.APPLY_TO_TAKE_SEAT) {  
        Log.i(TAG, "收到观众申请上麦请求: ${userInfo.userId}")  
    }  
}
```

#### Java

```
@Override  
public void onSeatRequestReceived(VoiceRoomDefine.RequestType type,  
    TUIRoomDefine.UserInfo userInfo) {  
    if (type == VoiceRoomDefine.RequestType.APPLY_TO_TAKE_SEAT) {  
        Log.i(TAG, "收到观众上麦请求: " + userInfo.userId);  
    }  
}
```



## iOS

## swift

```
import RTCRoomEngine
import LiveStreamCore

func onSeatRequestReceived(type: SGRequestType, userInfo: TUIUserInfo)
{
    if type == .applyToTakeSeat {
        print("收到观众申请上麦请求: \(userInfo.userId)")
    }
}
```

## Flutter

## Dart

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

// 创建监听Observer实例
final controller = SeatGridController();
final exampleObserver = ExampleObserver();

// 添加观察者
controller.addObserver(exampleObserver);

class ExampleObserver extends SeatGridWidgetObserver {
    ExampleObserver() {
        super.onSeatRequestReceived = (type, userInfo) {
            if (type == RequestType.applyToTakeSeat) {
                debugPrint('收到观众申请上麦请求: ${userInfo.userId}');
            }
        };
    }
}
```

```
}
```

**⚠ 注意：**

只有当房间模式是 APPLY\_TO\_TAKE (申请上麦)，房主才会收到上麦请求，FREE\_TO\_TAKE (自由上麦) 模式下，takeSeat 会直接上麦成功。

## 主播响应上麦请求

主播 A 收到观众的上麦请求后，可以调用 responseRemoteRequest 响应观众 B 是否同意上麦，观众 B 会收到主播 A 同意和拒绝（onAccepted/onRejected）的回调。

### Android

#### Kotlin

```
// 主播同意观众上麦
seatGridView.responseRemoteRequest(userId, true, null);

// 主播拒绝观众上麦
seatGridView.responseRemoteRequest(userId, false, null);
```

#### Java

```
// 主播同意观众上麦
seatGridView.responseRemoteRequest(userId, true, null);

// 主播拒绝观众上麦
seatGridView.responseRemoteRequest(userId, false, null);
```

### iOS

#### swift

```
import RTCRoomEngine
```

```
import LiveStreamCore

// 主播同意观众上麦
seatGridView.responseRemoteRequest(userId, true) {
} onError: { code, message in
}

// 主播拒绝观众上麦
seatGridView.responseRemoteRequest(userId, false) {
} onError: { code, message in
}
```

## Flutter

### Dart

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

final controller = SeatGridController();

const String userId = 'Replace with the userId of userB';

// 主播同意观众上麦
final result = await controller.responseRemoteRequest(userId, true);

// 主播拒绝观众上麦
final result = await controller.responseRemoteRequest(userId, false);
```

## 麦位信息发生变化的回调

### Android

如果您已经设置过 [自定义麦位视图](#)，可以监听 `updateSeatView` 回调来刷新自己的自定义 UI。

### Kotlin

```
override fun updateSeatView(seatGridView: SeatGridView, seatInfo:
TUIRoomDefine.SeatInfo, seatView: View) {
    Log.i(TAG, "麦位信息发生变化");
}
```

## Java

```
@Override
public void void updateSeatView(SeatGridView seatGridView,
TUIRoomDefine.SeatInfo seatInfo, View seatView) {
    Log.i(TAG, "麦位信息发生变化");
}
```

## iOS

如果您已经设置过 [自定义麦位视图](#)，可以监听updateSeatView回调来刷新自己的自定义 UI。

## swift

```
import RTCRoomEngine
import LiveStreamCore

func seatGridView(_ view: SeatGridView, updateSeatView seatInfo:
TUISeatInfo, seatView: UIView) {
    print("麦位信息发生变化")
}
```

## Flutter

当有麦位信息发生变化时，`SeatWidgetBuilder` 的 `seatInfoNotifier` 参数对应数值会发生变化，您可将您的自定义麦位部件作为 `ValueListenableBuilder` 的子部件，以便即时更新您的自定义麦位 UI。

## Dart

```
// seatWidgetBuilder 定义
typedef SeatWidgetBuilder = Widget Function(
```

```
BuildContext context,
ValueNotifier<TUISeatInfo> seatInfoNotifier,
ValueNotifier<int> volumeNotifier);

// 使用示例
import 'package:live_stream_core/live_stream_core.dart';
import 'package:rtc_room_engine/rtc_room_engine.dart';

SeatGridWidget (
  controller: controller,
  seatWidgetBuilder: (
    BuildContext context,
    ValueNotifier<TUISeatInfo> seatInfoNotifier,
    ValueNotifier<int> volumeNotifier) {
    return ValueListenableBuilder(
      valueListenable: seatInfoNotifier,
      builder: (context, seatInfo, _) {
        // 返回您的自定义麦位部件
        return Container();
      }
    );
  }
)
```

## 主播邀请观众上麦流程

### 主播发送邀请上麦请求

主播 A 给观众 C 发送邀请上麦请求，观众 C 会在 onSeatRequestReceived 回调中收到主播 A 的邀请上麦请求。

#### Android

##### Kotlin

```
val seatIndex = 1;
val userId = "userIdC";
val timeout = 60;
seatGridView.takeUserOnSeatByAdmin(seatIndex, timeout, userId, object
: VoiceRoomDefine.RequestCallback {
```

```
override fun onAccepted(userInfo: TUIRoomDefine.UserInfo) {
    Log.i(TAG, "邀请上麦被同意")
}

override fun onRejected(userInfo: TUIRoomDefine.UserInfo) {
    Log.i(TAG, "邀请上麦被拒绝")
}

override fun onCancelled(userInfo: TUIRoomDefine.UserInfo) {
    Log.i(TAG, "邀请上麦被取消")
}

override fun onTimeout(userInfo: TUIRoomDefine.UserInfo) {
    Log.i(TAG, "邀请上麦超时")
}

override fun onError(userInfo: TUIRoomDefine.UserInfo, error:
TUICommonDefine.Error, message: String) {
    Log.i(TAG, "邀请上麦错误")
}
})
```

## Java

```
val seatIndex = 1;
val userId = "userIdC";
val timeout = 60;
seatGridView.takeUserOnSeatByAdmin(seatIndex, userId, timeout, new
VoiceRoomDefine.RequestCallback() {
    @Override
    public void onAccepted(TUIRoomDefine.UserInfo userInfo) {
        Log.i(TAG, "邀请上麦被同意");
    }

    @Override
    public void onRejected(TUIRoomDefine.UserInfo userInfo) {
        Log.i(TAG, "邀请上麦被拒绝");
    }

    @Override
    public void onCancelled(TUIRoomDefine.UserInfo userInfo) {
        Log.i(TAG, "邀请上麦被取消");
    }
})
```

```
@Override
public void onTimeout(TUIRoomDefine.UserInfo userInfo) {
    Log.i(TAG, "邀请上麦超时");
}

@Override
public void onError(TUIRoomDefine.UserInfo userInfo,
TUICommonDefine.Error error, String message) {
    Log.i(TAG, "邀请上麦错误");
}
});
```

## iOS

### swift

```
import RTCRoomEngine
import LiveStreamCore

let seatIndex = 1
let timeout = 60
let userId = "userIdC"

seatGridView.takeUserOnSeatByAdmin(index: seatIndex, timeout: timeout,
userId: userId) { userInfo in
    print("邀请上麦被同意")
} onRejected: { userInfo in
    print("邀请上麦被拒绝")
} onCancelled: { userInfo in
    print("邀请上麦被取消")
} onTimeout: { userInfo in
    print("邀请上麦超时")
} onError: { userInfo, code, message in
    print("邀请上麦错误")
}
```

## Flutter

## Dart

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

const int seatIndex = 1;
const String userId = 'userIdC'
const int timeout = 60;
final controller = SeatGridController();
final result = await controller.takeUserOnSeatByAdmin(seatIndex,
userId, timeout);
if (result.code == TUIError.success) {
  switch (result.type) {
    case RequestResultType.onAccepted:
      debugPrint('邀请上麦被同意');
      break;
    case RequestResultType.onRejected:
      debugPrint('邀请上麦被拒绝');
      break;
    case RequestResultType.onCancelled:
      debugPrint('邀请上麦被取消');
      break;
    case RequestResultType.onTimeout:
      debugPrint('邀请上麦超时');
      break;
    default:
      break;
  }
} else {
  debugPrint('邀请上麦错误');
}
```

## 观众端收到邀请上麦请求

观众 C 会在 onSeatRequestReceived 回调方法中收到 主播A 的邀请上麦请求。

## Android



## Kotlin

```
override fun onSeatRequestReceived(type: VoiceRoomDefine.RequestType,
    userInfo: TUIRoomDefine.UserInfo) {
    if (type == VoiceRoomDefine.RequestType.INVITE_TO_TAKE_SEAT) {
        Log.i(TAG, "收到主播邀请上麦请求: ${userInfo.userId}")
    }
}
```

## Java

```
@Override
public void onSeatRequestReceived(VoiceRoomDefine.RequestType type,
    TUIRoomDefine.UserInfo userInfo) {
    if (type == VoiceRoomDefine.RequestType.INVITE_TO_TAKE_SEAT) {
        Log.i(TAG, "收到主播邀请上麦请求: " + userInfo.userId);
    }
}
```

## iOS

### swift

```
import RTCRoomEngine
import LiveStreamCore

func onSeatRequestReceived(type: SGRequestType, userInfo: TUIUserInfo)
{
    if type == .inviteToTakeSeat {
        print("收到主播邀请上麦请求: \(userInfo.userId)")
    }
}
```

## Flutter

## Dart

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

// 创建监听Observer实例
final controller = SeatGridController();
final exampleObserver = ExampleObserver();

// 添加观察者
controller.addObserver(exampleObserver);

class ExampleObserver extends SeatGridWidgetObserver {
  ExampleObserver() {
    super.onSeatRequestReceived = (type, userInfo) {
      if (type == RequestType.inviteToTakeSeat) {
        debugPrint('收到主播邀请上麦请求: ${userInfo.userId}');
      }
    };
  }
}
```

## 观众响应邀请上麦请求

观众 C 收到观众的上麦请求后，可以调用 `responseRemoteRequest` 响应主播 A 是否同意上麦，主播 A 会收到观众 C 同意和拒绝（`onAccepted/onRejected`）的回调。

## Android

### Kotlin

```
// 观众同意主播邀请
seatGridView.responseRemoteRequest("", true, null);

// 观众拒绝主播邀请
seatGridView.responseRemoteRequest("", false, null);
```

## Java

```
// 观众同意主播邀请
seatGridView.responseRemoteRequest("", true, null);

// 观众拒绝主播邀请
seatGridView.responseRemoteRequest("", false, null);
```

## iOS

### swift

```
import RTCRoomEngine
import LiveStreamCore

// 观众同意主播邀请
seatGridView.responseRemoteRequest("userId of anchor", true) {
} onError: { code, message in
}

// 观众拒绝主播邀请
seatGridView.responseRemoteRequest("userId of anchor", false, null) {
} onError: { code, message in
}
```

## Flutter

### Dart

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

final controller = SeatGridController();

const String userId = 'userId of anchor';
```

```
// 观众同意主播邀请
final result = await controller.responseRemoteRequest(userId, true);

// 观众拒绝主播邀请
final result = await controller.responseRemoteRequest(userId, false);
```

#### ❗ 说明:

如果自定义麦位视图，主播和观众都可以通过监听 [麦位信息发生变化的回调](#) 刷新麦位 UI。

## 下麦流程

### 观众上麦成功后，主动下麦

观众 B 上麦成功后，观众 B 主动下麦可以调用 `leaveSeat`。

#### Android

##### Kotlin

```
seatGridView.leaveSeat()
```

##### Java

```
seatGridView.leaveSeat();
```

#### iOS

##### swift

```
import RTCRoomEngine
import LiveStreamCore

seatGridView.leaveSeat() {
} onError: { code, message in
```

```
}
```

## Flutter

### Dart

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

final controller = SeatGridController();
final result = await controller.leaveSeat();
```

## 观众上麦成功后，主播踢观众下麦

观众 B 上麦成功后，主播 A 踢观众 B 下麦。

## Android

### Kotlin

```
val userId = "userIdB"
seatGridView.kickUserOffSeatByAdmin(userId, null)
```

### Java

```
String userId = "userIdB";
seatGridView.kickUserOffSeatByAdmin(userId, null);
```

## iOS

### swift

```
import RTCRoomEngine
import LiveStreamCore

let userId = "userIdB"
seatGridView.kickUserOffSeatByAdmin(userId) {
} onError: { code, message in
}
```

## Flutter

### Dart

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

const String userId = "userIdB";
final controller = SeatGridController();
final result = await controller.kickUserOffSeatByAdmin();
```

## 观众收到被主播踢下麦的回调

主播 A 踢观众 B 下麦后，观众 B 会收到 onKickedOffSeat 回调。

## Android

### Kotlin

```
override fun onKickedOffSeat(inviterUser: UserInfo) {
    Log.i(TAG, "主播踢下麦")
}
```

### Java

```
@Override
public void onKickedOffSeat(TUIRoomDefine.UserInfo userInfo) {
```

```
Log.i(TAG, "主播踢下麦");  
}
```

## iOS

### swift

```
import RTCRoomEngine  
import LiveStreamCore  
  
func onKickedOffSeat(userInfo: TUIUserInfo) {  
    print("主播踢下麦")  
}
```

## Flutter

### Dart

```
import 'package:live_stream_core/live_stream_core.dart';  
import 'package:rtc_room_engine/rtc_room_engine.dart';  
  
// 创建监听Observer实例  
final controller = SeatGridController();  
final exampleObserver = ExampleObserver();  
  
// 添加观察者  
controller.addObserver(exampleObserver);  
  
class ExampleObserver extends SeatGridWidgetObserver {  
    ExampleObserver() {  
        super.onKickedOffSeat = (userInfo) {  
            debugPrint('主播踢下麦');  
        };  
    }  
}
```

```
}
```

### ❗ 说明:

如果自定义麦位视图，主播和观众都可以通过监听 [麦位信息发生变化的回调](#) 刷新麦位 UI。

## 锁麦流程

### 位置锁定

主播可将某个麦位进行锁定，此麦位就会被封禁，此麦位就无法进行上下麦操作。

#### Android

##### Kotlin

```
val index = 1;
val isLockSeat = true
val params = TUIRoomDefine.SeatLockParams().apply {
    lockSeat = isLockSeat
}
seatGridView.lockSeat(index, params, null)
```

##### Java

```
int index = 1;
bool isLockSeat = true;
TUIRoomDefine.SeatLockParams params = new
TUIRoomDefine.SeatLockParams();
params.lockSeat = isLockSeat;
seatGridView.lockSeat(index, params, null);
```

#### iOS

##### swift



```
import RTCRoomEngine
import LiveStreamCore

let index = 1
let isLockSeat = true

let params = TUISeatLockParams()
params.lockSeat = isLockSeat

seatGridView.lockSeat(index: index, lockMode: params) {
} onError: { code, message in
}
```

## Flutter

### Dart

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

const int index = 1;
const bool isLockSeat = true
final params = TUISeatLockParams();
params.lockSeat = isLockSeat;
final controller = SeatGridController();
final result = await controller.lockSeat(index, params);
```

## 音频锁定

主播可将某个麦位进行音频锁定，此麦位上的用户就会被静音。

## Android

### Kotlin

```
val index = 1;
bool isAudioLocked = true;
val params = TUIRoomDefine.SeatLockParams().apply {
    isAudioLocked = isLockSeat
}
seatGridView.lockSeat(index, params, null)
```

## Java

```
int index = 1;
bool isAudioLocked = true;
TUIRoomDefine.SeatLockParams params = new
TUIRoomDefine.SeatLockParams();
params.lockAudio = seatInfo.isAudioLocked;
seatGridView.lockSeat(index, params, null);
```

## iOS

### swift

```
import RTCRoomEngine
import LiveStreamCore

let index = 1
let isAudioLocked = true

let params = TUISeatLockParams()
params.lockAudio = isAudioLocked

seatGridView.lockSeat(index: index, lockMode: params) {
} onError: { code, message in
}
```

## Flutter

## Dart

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

const int index = 1;
const bool isAudioLocked = true
final params = TUISeatLockParams();
params.lockAudio = isAudioLocked;
final controller = SeatGridController();
final result = await controller.lockSeat(index, params);
```

### ❗ 说明:

如果自定义麦位视图，主播和观众都可以通过监听 [麦位信息发生变化的回调](#) 刷新麦位 UI。

## 其它麦位管理功能扩展

### Android

麦位管理功能是基于 `SeatGridView` 实现，若您需要扩展麦位管理功能，可参见 [SeatGridView](#) 文档。

### iOS

麦位管理功能是基于 `SeatGridView` 实现，若您需要扩展麦位管理功能，可参见 [SeatGridView](#) 文档。

### Flutter

麦位管理功能是基于 `SeatGridView` 实现，若您需要扩展麦位管理功能，可参见 [SeatGridWidget](#) 文档。

# 房间列表（TUILiveKit）

最近更新时间：2025-04-14 18:02:42

## 功能介绍

TUILiveKit 支持房间列表 UI 组件，房间列表组件可以帮您展示当前应用下所有的在线直播和语音聊天室的直播间，当您接入房间列表 UI 组件后，只需要点击列表中的某个直播间，就可以实时观看当前主播的直播内容。进入直播间后您还可以通过弹幕、礼物、连麦等功能和主播进行实时互动。

房间列表组件

观看直播

和主播连麦

## 功能接入

|         |
|---------|
| Android |
|         |

**❗ 说明:**

请务必确保已经按照 [快速接入（TUILiveKit）](#) 完成登录操作。只有 `TUILogin.login` 登录成功后才能正常进入直播预览画面。

**1. 在需要显示房间列表 UI 组件的 xml 中添加 FrameLayout 布局。**

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/fl_live_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

**2. 通过加载 TUILiveKit 的 `TUILiveListFragment` 页面到 xml 中定义的布局上，就可以展示房间列表。****Java**

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.app_activity_main);

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
        TUILiveListFragment listFragment = new TUILiveListFragment();
        fragmentTransaction.add(R.id.fl_live_list, listFragment);
        fragmentTransaction.commit();
    }
}
```

**iOS**

**说明：**

请务必确保已经按照 [快速接入](#) 完成登录操作。只有 `TUILogin.login` 登录成功后才能正常进入直播预览画面。

**Swift**

```
//  
//  MainViewController.swift  
//  
  
import UIKit  
import TUILiveKit  
  
@objc private func buttonTapped(_ sender: UIButton) {  
    // 进入房间列表  
    let liveListViewController = TUILiveListViewController()  
    self.navigationController?.pushViewController(viewController,  
animated: true)  
}
```

**Objective-C**

```
//  
//  MainViewController.m  
//  
  
#import <TUILiveKit/TUILiveKit-Swift.h>  
  
- (void)buttonTapped:(UIButton *)sender {  
    // 进入房间列表  
    TUILiveListViewController *liveListViewController =  
[[TUILiveListViewController alloc] init];  
    [self.navigationController  
pushViewController:liveListViewController animated:true];  
}
```

**Flutter**

**说明：**

请务必确保已经按照 [快速接入（TUILiveKit）](#) 完成登录操作。只有登录成功后才能正常进入直播预览画面。

在您的 widget 中，通过加载 TUILiveKit 的 `LiveListWidget` 组件，就可以展示房间列表。

**Dart**

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

return Scaffold(
  body: SizedBox(
    width: _screenWidth,
    height: double.infinity,
    child: LiveListWidget(), // 在自己的 widget 树中 添加 TUILiveKit 的
    房间列表组件 LiveListWidget
  ),
);
```

## 功能定制

如果当前的 UI 不满足您的需求，您可以通过修改如下目录中的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对房间列表相关的文件做了介绍。

**Android**

```
// 文件位置：
Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/common/uicomponent/

roomlist // 直播间列表组件的实现目录
├── service // 直播间列表组件 的服务层目录
│   └── RoomListService.java // 直播间列表组件服务层的具体实现，封装了直播间列表相关的 api
```

```

├─ store                                // 直播间列表组件 的数据层目
录
|   └─ RoomListState.java              // 直播间列表组件 数据的具体
封装类
|   └─ view                             // 直播间列表组件 的视图层目
录
|   └─ adapter                          // 直播间列表组件 的视图层的
适配器目录
|       └─ LoadMoreAdapterWrapper.java // 直播间列表组件 对直播间列
表适配器扩展了下拉加载功能的适配器
|       └─ RoomListAdapter.java         // 直播间列表组件 的直播间列
表适配器
|   └─ ListAudienceActivity.java       // 直播间列表组件 中点击某一
个直播间，拉起的直播拉流页面
|   └─ RoomListView.java                // 直播间列表组件 直播间列表
视图的实现

```

## iOS

```

// 文件位置: iOS/TUILiveKit/Sources/Component/LiveList/

├─ LiveList                            // 直播间列表组件的实现目录
|   └─ Service                          // 直播间列表组件 的服务层目录
|       └─ LiveListService.swift        // 直播间列表组件服务层的具体
实现，封装了直播间列表相关的 api
|       └─ Store                        // 直播间列表组件 的数据层目录
|           └─ LiveListActions.swift     // 直播间列表组件 的事件定义
类，定义了所有直播列表相关事件
|           └─ LiveListReducer.swift     // 直播间列表组件 的事件响应
类，当有事件发生时触发，用于监听修改直播列表相关数据
|           └─ LiveListSelectors.swift   // 直播间列表组件 的数据选择
类，获取数据源的值
|           └─ LiveListState.swift       // 直播间列表组件 的数据定义
类，定义了所有与直播列表有关的数据模型
|       └─ LiveListStore.swift           // 直播间列表组件 的数据驱动和
事件发送协议类
|   └─ View                             // 直播间列表组件 的视图层目录
|       └─ LiveListCell.swift            // 直播间列表组件 的自定义
Cell

```



```
| └─ LiveListRootView.swift // 直播间列表组件 的根视图
```

## Flutter

```
// 文件位置: TUILiveKit/Flutter/livekit/lib/component/  
  
room_list // 直播间列表组件的实现目  
录  
└─ service // 直播间列表组件 的服务  
层目录  
└─ room_list_service.dart // 直播间列表组件服务层的  
具体实现, 封装了直播间列表相关的 api  
└─ store // 直播间列表组件 的数据  
层目录  
└─ room_list_state.dart // 直播间列表组件 数据的  
具体封装类  
└─ view // 直播间列表组件 的视图  
层目录  
└─ room_list_widget.dart // 直播间列表组件 直播间  
列表视图的实现
```

## 关键代码

### 获取直播间列表插件

## Android

### Java

```
// 文件位置:  
Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/common/uicompo  
nent/roomlist/service/RoomListService.java  
  
private final TUILiveListManager mTUILiveListManager ;
```

```
mTUILiveListManager = ( TUILiveListManager )
TUIRoomEngine . sharedInstance ( ) . getExtension ( LIVE_LIST_MANAGER ) ;
```

## iOS

### Swift

```
// 文件位置：
iOS/TUILiveKit/Source/Component/LiveList/Service/LiveListService.swift

let listManager = roomEngine.getExtension(extensionType:
.liveListManager) as? TUILiveListManager
```

## Flutter

### Dart

```
// 文件位置：
TUILiveKit/Flutter/livekit/lib/component/room_list/service/room_list_
service.dart

late final TUILiveListManager _liveListManager =
TUIRoomEngine.sharedInstance().getExtension(TUIExtensionType.liveListM
anager);
```

## 获取直播间列表

## Android

### Java

// 文件位置:

```
Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/common/uicomponent/roomlist/service/RoomListService.java
```

```
private static final int FETCH_LIST_COUNT = 20;
public String cursor = "";
mTUILiveListManager.fetchLiveList(cursor, FETCH_LIST_COUNT, new
LiveInfoListCallback() {
    @Override
    public void onSuccess(LiveInfoListResult result) {
    }

    @Override
    public void onError(TUICommonDefine.Error error, String s) {
    }
});
```

## iOS

### Swift

// 文件位置:

```
iOS/TUILiveKit/Source/Component/LiveList/Service/LiveListService.swift
```

```
func getLiveList(cursor: String, count: Int = 20) ->
AnyPublisher<(String, [TUILiveInfo]), InternalError> {
    return Future<(String, [TUILiveInfo]), InternalError> { [weak self]
promise in
    guard let self = self else { return }
    guard let listManager = roomEngine.getExtension(extensionType:
.liveListManager) as? TUILiveListManager else {
        promise(.failure(InternalError(error:TUIError.failed, message:
"getRoomListFailed")))
        return
    }
    listManager.fetchLiveList(cursor: cursor, count: count) {
cursor, liveInfoList in
        promise(.success((cursor, liveInfoList)))
    } onError: { error, message in
```

```
        promise(.failure(InternalError(error: error, message:
message)))
    }
    }.eraseToAnyPublisher()
}
```

## Flutter

### Dart

```
// 文件位置:
TUILiveKit/Flutter/lib/component/room_list/service/room_list_service.
dart

Future<void> _fetchLiveList() async {
    final String cursor = roomListState.cursor;
    TUIValueCallBack<TUILiveListResult> result = await
_liveListManager.fetchLiveList(cursor, fetchListCount);
    if (result.code != TUIError.success) {
        ErrorHandler.onError(result.code);
        roomListState.loadStatus.value = false;
        roomListState.refreshStatus.value = false;
        roomListState.isHaveMoreData.value = false;
    } else {
        final liveListResult = result.data as TUILiveListResult;
        roomListState.liveInfoList.value = liveListResult.liveInfoList;
        roomListState.cursor = liveListResult.cursor;
        roomListState.loadStatus.value = false;
        roomListState.refreshStatus.value = false;
        roomListState.isHaveMoreData.value =
liveListResult.cursor.isNotEmpty;
    }
}
```

# 关注主播（TUILiveKit）

最近更新时间：2025-04-03 17:08:42

## 功能介绍

在直播场景下，关注主播功能是一种重要的互动方式，它允许观众选择关注特定的主播，从而在直播平台上建立长期的互动和关注关系。TUILiveKit 中已经通过即时通信 IM 实现了关注主播的功能。

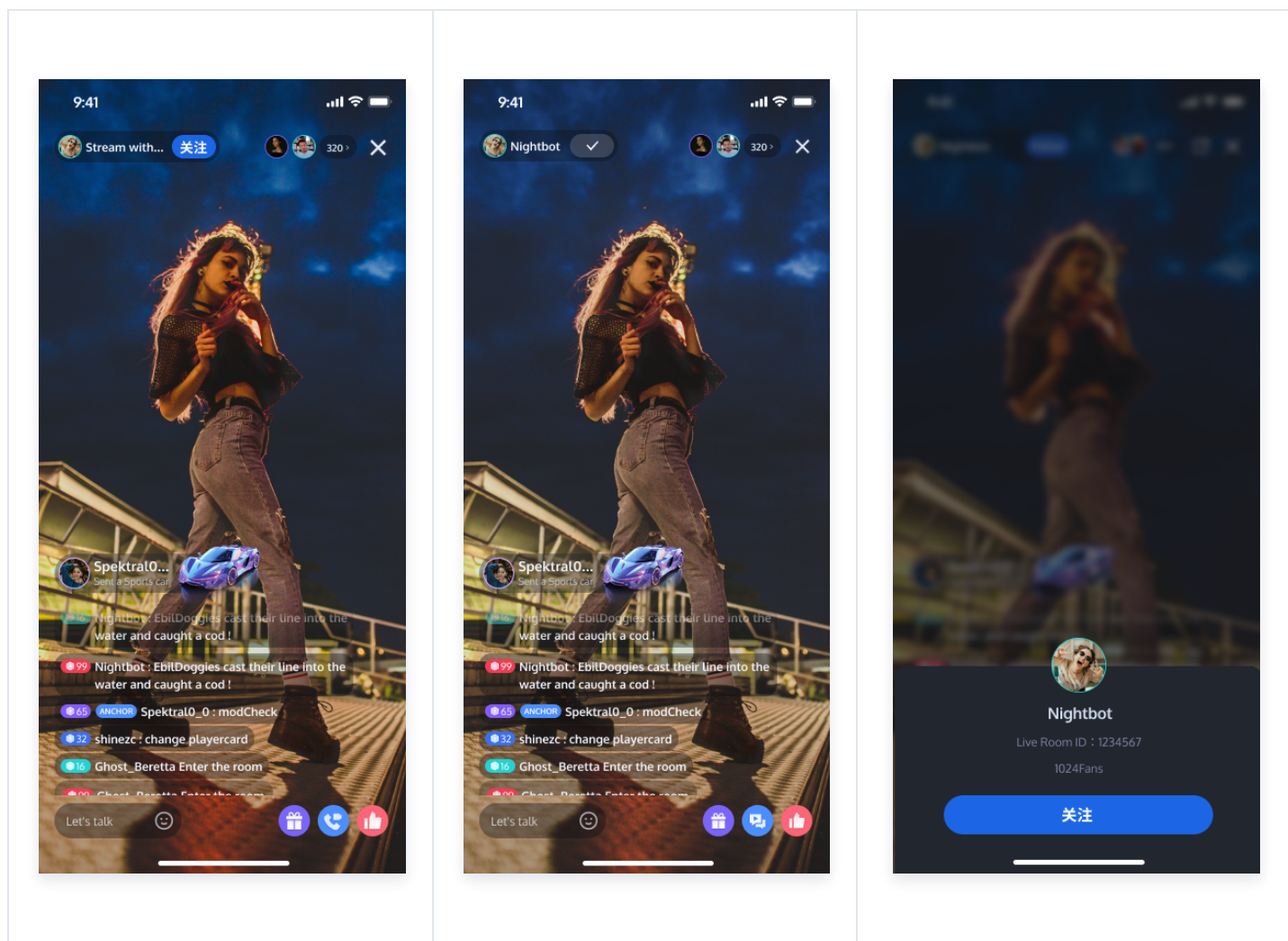
- **关注** 功能允许用户选择他们感兴趣的其他用户，以便及时获取这些用户最新的动态、发布或活动信息，您可以根据用户的关注列表提供个性化的内容推荐。
- **粉丝** 功能是指用户被其他人关注的状态，当用户 A 关注用户 B 时，A 就成为了 B 的粉丝。

通过关注和粉丝功能，社交应用和网站能够创建一个活跃的、相互连接的用户网络，促进信息的传播和社区的构建。

## 使用说明

**说明：**  
TUILiveKit 的**关注主播**功能依赖于**即时通信 IM 的关注&粉丝相关的 API**，所以需要开通**即时通信 IM 相关**的套餐包。具体可参见 [关注&粉丝](#) 文档。

| 未关注状态 | 关注状态 | 主播信息面板 |
|-------|------|--------|
|-------|------|--------|



- **关注主播：**您可以在直播界面的左上角的直播间信息区域点击 **关注** 按钮关注主播
- **取消关注：**您可以在直播界面的左上角的直播间信息区域点击 **取消关注** 按钮可以取消关注主播
- **查看主播粉丝数：**您可以点击直播界面左上角的直播间信息区域弹出直播间详情面板，该面板会展示主播的**粉丝数量**。

## 功能定制

如果当前的 UI 不满足您的需求，您可以通过修改如下目录中的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对关注功能相关的文件做了介绍。

### Android

```
// 文件位置：  
Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/common/uicomponent/
```

```
roominfo // 直播间信息UI组件的实现目录
├─ RoomInfoDetailView.java // 直播间信息详情面板视图的具体实现
├─ RoomInfoView.java // 直播间信息视图的具体实现，点击该视图，会展示详情面板视图
```

## iOS

```
// 文件位置: TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/

LiveInfo // 直播间信息UI组件的实现目录
├─ LiveInfoView.swift // 直播间信息视图的具体实现，点击该视图，会展示详情面板视图
├─ View
└─ LiveInfoPanelView.swift // 直播间信息详情面板视图的具体实现
```

## 关键代码

### 关注主播

## Android

### Java

```
// 文件位置:
Android/tuilivekit/src/Main/java/com/trtc/uikit/livekit/service/impl/
LiveServiceImpl.java

List<String> userIDList = new ArrayList<>();
userIDList.add("userId");

V2TIMManager.getFriendshipManager().followUser(userIDList,
    new V2TIMValueCallback<List<V2TIMFollowOperationResult>>() {
        @Override
        public void onSuccess(List<V2TIMFollowOperationResult>
results) {
        }
    })
```

```
@Override
public void onError(int code, String message) {
}
});
```

## iOS

### Swift

// 文件位置:

TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoService

```
public func followUser(userId: String) {
    let userInfo = TUIUserInfo()
    userInfo.userId = userId
    V2TIMManager.sharedInstance().followUser([userId]) { [weak self]
followResultList in
        guard let self = self, let result = followResultList?.first
else { return }
        if result.resultCode == 0,
!self.state.followingList.contains(where: { $0.userId == userId }) {
            self.state.followingList.insert(userInfo)
            self.getFansNumber()
        }
    } fail: { code, message in
        debugPrint("[RoomInfo] followUser failed, error:\(code),
message:\(String(describing: message))")
    }
}
```

## 取消关注主播

### Android



## Java

// 文件位置:

```
Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/service/impl/  
LiveServiceImpl.java
```

```
List<String> userIDList = new ArrayList<>();  
userIDList.add("userId");
```

```
V2TIMManager.getFriendshipManager().unfollowUser(userIDList,  
    new V2TIMValueCallback<List<V2TIMFollowOperationResult>>() {  
        @Override  
        public void onSuccess(List<V2TIMFollowOperationResult>  
results) {  
            }  
  
        @Override  
        public void onError(int code, String message) {  
            }  
    });
```

## iOS

### Swift

// 文件位置:

```
TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoService
```

```
public func unfollowUser(userId: String) {  
    V2TIMManager.sharedInstance().unfollowUser([userId]) { [weak self]  
followResultList in  
        guard let self = self, let result = followResultList?.first  
else { return }  
        if result.resultCode == 0 {  
            self.state.followingList = state.followingList.filter {  
$0.userId != userId}  
        }  
    }
```

```
    } fail: { code, message in
        debugPrint("[RoomInfo] unfollowUser failed, error:\(code),
message:\(String(describing: message))")
    }
}
```

## 查看主播关注状态

### Android

#### Java

// 文件位置:

Android/tuiliblivekit/src/Main/java/com/trtc/uikit/livekit/service/impl/  
LiveServiceImpl.java

```
List<String> userIDList = new ArrayList<>();
userIDList.add("userId");
```

```
V2TIMManager.getFriendshipManager().checkFollowType(userIDList,
    new V2TIMValueCallback<List<V2TIMFollowTypeCheckResult>>() {
        @Override
        public void onSuccess(List<V2TIMFollowTypeCheckResult>
results) {
        }

        @Override
        public void onError(int code, String message) {
        }
    });
```

### IOS

// 文件位置:

TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoService

```
public func isFollow(userId: String) {
    let userInfo = TUIUserInfo()
    userInfo.userId = userId
    V2TIMManager.sharedInstance().checkFollowType([state.ownerId]) {
[weak self] checkResultList in
        guard let self = self, let result = checkResultList?.first
    else { return }
        if result.followType == .FOLLOW_TYPE_IN_BOTH_FOLLOWERS_LIST ||
result.followType == .FOLLOW_TYPE_IN_MY_FOLLOWING_LIST {
            if !self.state.followingList.contains(where: { $0.userId
== userId }) {
                self.state.followingList.insert(userInfo)
            }
        } else {
            self.state.followingList = state.followingList.filter {
$0.userId != userId}
        }
    } fail: { code, message in
        debugPrint("[RoomInfo] isFollow failed, error:\(code),
message:\(String(describing: message))")
    }
}
```

## 获取粉丝数量

### Android

#### Java

```
// 文件位置:
Android/tuillivekit/src/Main/java/com/trtc/uikit/livekit/service/impl/
LiveServiceImpl.java

List<String> userIDList = new ArrayList<>();
userIDList.add("userId");

V2TIMManager.getFriendshipManager().getUserFollowInfo(userIDList,
    new V2TIMValueCallback<List<V2TIMFollowInfo>>() {
        @Override
        public void onSuccess(List<V2TIMFollowInfo> results) {
```

```
    }

    @Override
    public void onError(int code, String message) {
    }

});
```

## iOS

```
// 文件位置:
TUILiveKit/iOS/TUILiveKit/Sources/Component/LiveInfo/RoomInfoService

public func getFansNumber() {
    V2TIMManager.sharedInstance().getUserFollowInfo([state.ownerId]) {
[weak self] followInfoList in
    guard let self = self, let followInfo = followInfoList?.first
else { return }
    self.state.fansNumber = Int(followInfo.followersCount)
    } fail: { code, message in
        debugPrint("[RoomInfo] getFansNumber failed, error:\(code),
message:\(String(describing: message))")
    }
}
```

# 互动弹幕（TUILiveKit）

最近更新时间：2025-07-23 10:01:22

## 功能介绍

互动弹幕功能是一项重要的实时交流工具，支持多种互动方式。用户可以在弹幕中输入表情和文本消息，增强消息的趣味性，使得互动体验更加愉快和生动。通过这一功能，观众能够在直播过程中与主播和其他观众进行更为丰富的交流，提升整体的参与感和乐趣。TUILiveKit 中已经通过即时通信 IM 实现了互动弹幕的功能。

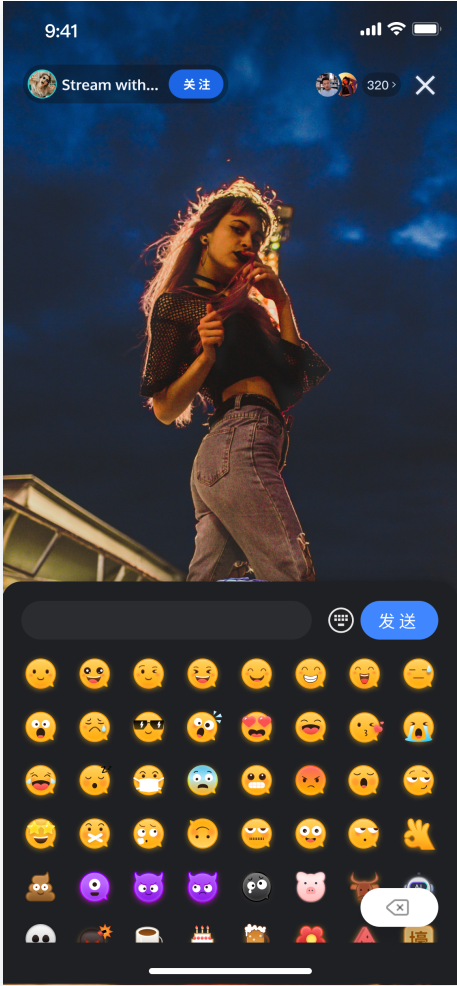
iOS

弹幕组件提供 2 个视图组件：

- `BarrageInputView`：发送弹幕消息视图，点击后可以拉起输入界面。
- `BarrageStreamView`：接收弹幕消息视图，会显示弹幕消息到该视图上。

效果如下图所示：

| 展示 | 输入 |
|----|----|
|    |    |



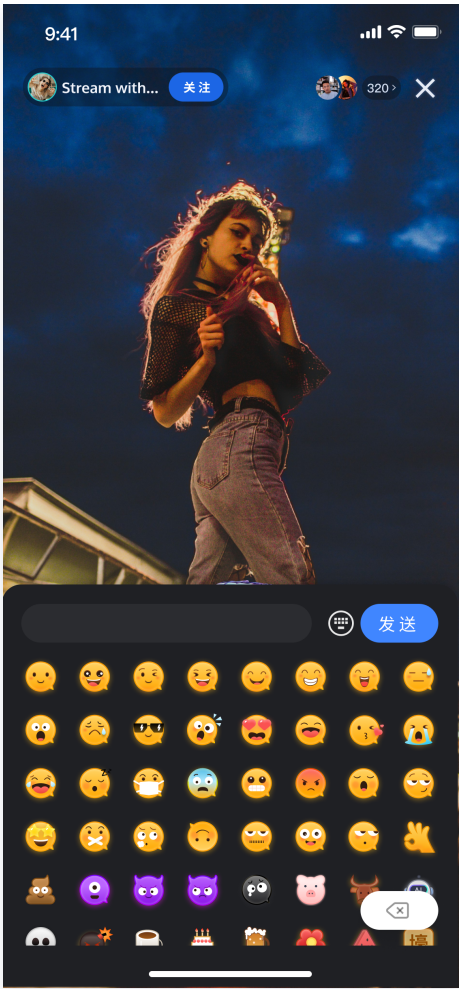
Flutter

弹幕组件主要包提供2个 Widget 组件，发送弹幕消息 Widget ( `BarrageSendWidget` ) 和显示弹幕消息的 Widget ( `BarrageDisplayWidget` )：

- `BarrageSendWidget` ：发送弹幕消息 Widget，点击后可以拉起输入界面。
- `BarrageDisplayWidget` ：接收到弹幕消息后，会显示弹幕消息到该 Widget 上。

效果如下图所示：

| 展示 | 输入 |
|----|----|
|    |    |



Android

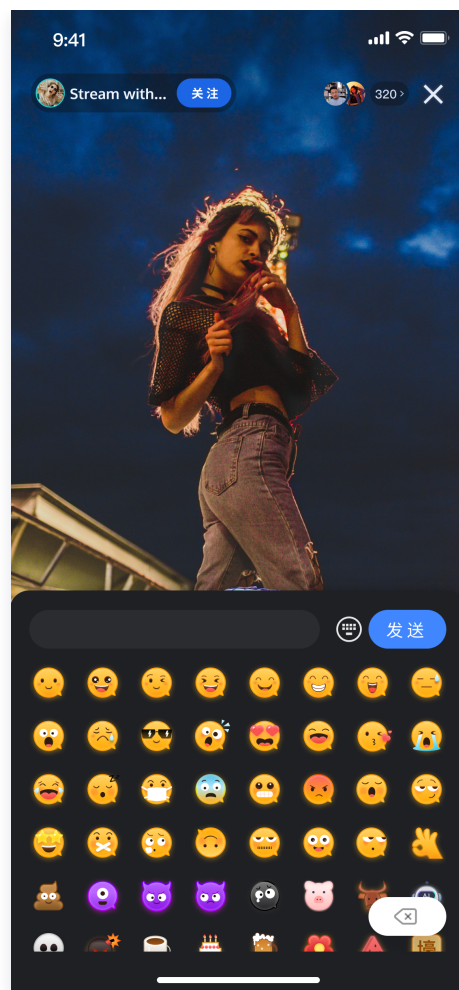
弹幕组件提供 2 个视图组件：

- `BarrageInputView` ： 发送弹幕消息视图，点击后可以拉起输入界面。
- `BarrageStreamView` ： 接收弹幕消息视图，会显示弹幕消息到该视图上。

效果如下图所示：

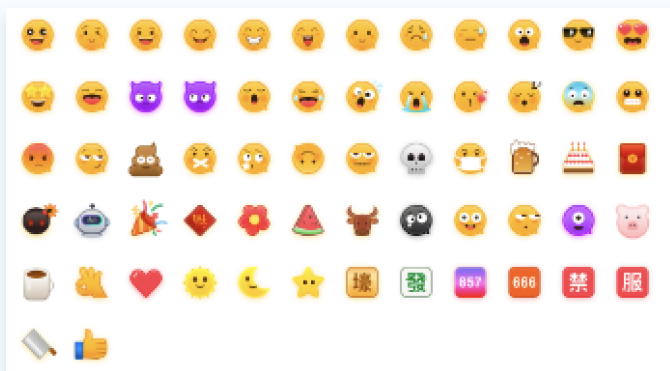
| 展示 | 输入 |
|----|----|
|    |    |





### 说明：

1. 支持系统键盘和表情键盘切换。
2. 为尊重表情设计版权，TUILiveKit 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。下图所示默认的小黄脸表情包版权归腾讯云所有，可有偿授权使用，如需获得授权可 [提交工单](#) 联系我们。



## 快速接入



**说明：**

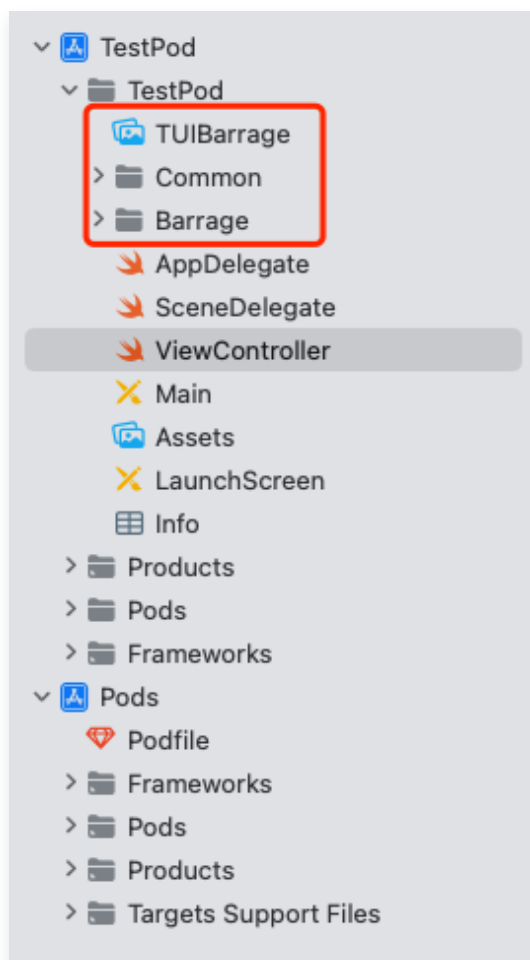
如果您已经集成了 LiveKit 组件，我们在 LiveKit 中默认已经接入了 barrage 组件，您可以直接在 LiveKit 中体验弹幕功能。

## 添加 barrage 组件到项目中

### iOS

1. 从 [GitHub](#) 下载 Barrage 组件到本地，并将 Barrage (TUILiveKit/Sources/Component/Barrage) 目录拷贝到您的工程中；
2. 拷贝依赖组件 Common (TUILiveKit/Sources/Common) 到您的工程中；
3. 拷贝资源文件 TUIBarrage.xcassets (TUILiveKit/Resources/TUIBarrage.xcassets) 到您的工程中。
4. 拷贝语言文件 Localized (TUILiveKit/Resources/Localized) 到您的工程中。

目录如下图所示：



5. 在您的Podfile文件中，添加如下依赖：

- `pod 'RTCCCommon'`

- pod 'TUICore'
- pod 'SnapKit'
- pod 'RTCRoomEngine'

如图：

```
4
5 platform :ios, '13.0'
6
7 target 'TestPod' do
8   # Comment the next line if you don't want to use dynamic frameworks
9   use_frameworks!
10
11   pod 'RTCCCommon'
12   pod 'TUICore'
13   pod 'SnapKit'
14   pod 'RTCRoomEngine'
15
16 end
17
```

## 6. 执行 pod install

## Flutter

1. 在项目工程的 pubspec.yaml 文件的 dependencies 节点中，添加 **barrage** 的依赖。

```
dependencies:
  flutter:
    sdk: flutter
  flutter_localizations:
    sdk: flutter
  intl: ^0.19.0
  # 添加 barrage 依赖
  live_uikit_barrage: ^1.0.0
```

2. 执行 `flutter pub get` 命令
3. 配置多语言支持，在您应用 `MaterialApp` 的 `localizationsDelegates` 和 `supportedLocales` 属性上添加 `gift` 组件的多语言支持。

```
MaterialApp(localizationsDelegates: const [
  ...BarrageLocalizations.localizationsDelegates,
], supportedLocales: const [
  ...BarrageLocalizations.supportedLocales,
],
// ...
);
```

## Android

1. 从 [GitHub](#) 下载 TUILiveKit 到本地。
2. 将 Android/tuilivekit/component/barrage 目录拷贝到您的工程中。
3. 将 Android/tuilivekit/component/common 目录拷贝到您的工程中。
4. 并在您工程的settings.gradle文件中增加配置：

```
include ':common'
include ':barrage'
```

5. 在您需要引入的工程(build.gradle)文件中，增加依赖配置：

```
api project(':common')
api project(':barrage')
```

## 接入发送弹幕消息组件

### iOS

在需要发送弹幕的位置，创建 `BarrageInputView` ，点击后可以拉起输入界面：

```
let barrageInputView = BarrageInputView(roomId: roomId)
addSubview(barrageInputView)
```

### Flutter

在您需要接入发送弹幕消息的地方构建 `BarrageSendController` 和 `BarrageSendWidget` 对象，并将构建的 `BarrageSendWidget` 对象添加到您的 `Widget` 树上。示例代码如下：

```
BarrageSendController _sendController = BarrageSendController(  
    roomId: "liveRoomId",          /// liveRoomId 替  
    换为您的直播间ID  
    ownerId: "liveOwnerId",        /// liveOwnerId 替  
    换为您的直播间主播ID  
    selfUserId: "selfUserId",      /// selfUserId 替  
    换为您的当前登录的用户ID  
    selfName: "selfUserName";      /// selfUserName 替  
    换为您的当前登录的用户昵称  
BarrageSendWidget(controller: _sendController);
```

## Android

在需要发送弹幕的位置，创建 `BarrageInputView`，点击后可以拉起输入界面：

```
BarrageInputView barrageInputView = new BarrageInputView(mContext);  
barrageInputView.init(roomId);  
mBarrageInputContainer.addView(barrageInputView);
```

## 接入显示弹幕消息组件

### iOS

在需要展示弹幕的位置，使用 `BarrageStreamView` 来展示弹幕消息：

```
let barrageDisplayView = BarrageStreamView(roomId: roomId)  
addSubview(barrageDisplayView)
```

在获取到房主信息后，设置 `ownerId` 用来区分房主与观众的显示效果。

```
barrageDisplayView.setOwnerId(ownerId)
```

## Flutter

在您需要接入显示弹幕消息的地方构建 `BarrageDisplayController` 和 `BarrageDisplayWidget` 对象，并将构建的 `BarrageDisplayWidget` 对象添加到您的 Widget 树上。示例代码如下：

```
BarrageDisplayController _displayController =  
BarrageDisplayController(  
    roomId: "liveRoomId",           /// liveRoomId 替  
    换为您的直播间ID  
    ownerId: "liveOwnerId",         /// liveOwnerId 替  
    换为您的直播间主播ID  
    selfUserId: "selfUserId",       /// selfUserId 替  
    换为您的当前登录的用户ID  
    selfName: "selfUserName";       /// selfUserName 替  
    换为您的当前登录的用户昵称  
BarrageDisplayWidget(controller: _displayController);
```

## Android

在需要展示弹幕的位置，使用 `BarrageStreamView` 来展示弹幕消息：

```
BarrageStreamView barrageStreamView = new BarrageStreamView(mContext);  
// ownerId 表示房主的 ID，用来区分房主与观众的显示效果  
barrageStreamView.init(roomId, ownerId);  
mLayoutBarrageContainer.addView(barrageStreamView);
```

## 插入本地弹幕消息

### iOS

弹幕展示组件 `BarrageStreamView` 对外提供 `insertBarrages` 接口方法，用于（批量）插入自定义消息（如 礼物消息、直播间公告），通常自定义消息配合自定义样式，可以实现不一样的展示效果。

示例代码：

```
// 示例1：在弹幕区插入一条礼物消息  
let barrage = TUIBarrage()  
barrage.content = "gift"
```

```
barrage.user.userId = sender.userId
barrage.user.userName = sender.userName
barrage.user.avatarUrl = sender.avatarUrl
barrage.user.level = sender.level

barrage.extInfo["TYPE"] = AnyCodable("GIFTMESSAGE")
barrage.extInfo["gift_name"] = AnyCodable(gift.giftName)
barrage.extInfo["gift_count"] = AnyCodable(giftCount)
barrage.extInfo["gift_icon_url"] = AnyCodable(gift.imageUrl)
barrage.extInfo["gift_receiver_username"] =
AnyCodable(receiver.userName)

barrageDisplayView.insertBarrages([barrage])

// 示例2:发送弹幕后回显消息
// 实现 BarrageInputView 的代理 BarrageInputViewDelegate
extension MyViewController: BarrageInputViewDelegate {
    func barrageInputViewOnSendBarrage(_ barrage: TUIBarrage) {
        barrageDisplayView.insertBarrages([barrage])
    }
}
```

#### ❗ 说明:

TUIBarrage 的 extInfo 是一个 Map，用于存放自定义数据。

## Flutter

当您需要插入本地的弹幕消息时，可以调用 BarrageDisplayWidget 的 insertMessage 方法，插入本地消息。例如，LiveKit 中监听到观众进房后，可以插入一条观众进房的弹幕消息，示例代码如下所示：

```
BarrageUser barrageUser = BarrageUser();
barrageUser.userId = "enterRoomUserId";           /// 弹幕上展示的用户
Id信息
barrageUser.userName = "enterRoomUserName";        /// 弹幕上展示的用户昵
称信息
barrageUser.avatarUrl = "enterRoomUserAvatar";     /// 弹幕上展示的用户头
像信息
barrageUser.level = "66";                          /// 弹幕上展示的用户等
级信息
```

```
Barrage barrage = Barrage();
barrage.user = barrageUser;
barrage.content = "进入房间"; // 弹幕上展示的文本内容
_displayController.insertMessage(barrage);
```

## Android

弹幕展示组件 `BarrageStreamView` 对外提供 `insertBarrages` 接口方法，用于（批量）插入自定义消息（如 礼物消息、直播间公告），通常自定义消息配合自定义样式，可以实现不一样的展示效果。

示例代码：

```
// 在弹幕区插入一条礼物消息
Barrage barrage = new Barrage();
barrage.content = "gift";
barrage.user.userId = sender.userId;
barrage.user.userName = sender.userName;
barrage.user.avatarUrl = sender.avatarUrl;
barrage.user.level = sender.level;
barrage.extInfo.put(Constants.GIFT_VIEW_TYPE, GIFT_VIEW_TYPE_1);
barrage.extInfo.put(GIFT_NAME, barrage.giftName);
barrage.extInfo.put(GIFT_COUNT, giftCount);
barrage.extInfo.put(GIFT_ICON_URL, barrage.imageUrl);
barrage.extInfo.put(GIFT_RECEIVER_USERNAME, receiver.userName);

mBarrageStreamView.insertBarrages(barrage);
```

### ❗ 说明：

`Barrage` 的 `extInfo` 是一个 `Map`，用于存放自定义数据。

## 自定义弹幕消息

### iOS

弹幕消息样式默认有两种：普通弹幕消息样式 和 自定义消息样式。

如果您需要更多消息样式（如礼物发送回显），可实现 `BarrageStreamView` 的代理方法 `BarrageStreamViewDelegate`：

```
class MyViewController: BarrageStreamViewDelegate {
    func barrageDisplayView(_ barrageDisplayView: BarrageStreamView,
        createCustomCell barrage: TUIBarrage) -> UIView? {
        // 是否需要使用自定义UI，如不需要返回 nil 即可
        guard let type = barrage.extInfo["TYPE"], type.value as?
String == "GIFTMESSAGE" else {
            return nil
        }
        // 返回自定义消息样式UI（礼物回显）
        return CustomBarrageCell(barrage: barrage)
    }
}
```

## Flutter

当您需要针对特定的弹幕消息，展示自定义的弹幕 item 时，您可以通过 `BarrageDisplayWidget` 的 `setCustomBarrageBuilder` 方法实现。例如，自定义一个显示红色文本的弹幕示例代码如下所示：

```
/// 1、定义一个自定义弹幕 item 构建器
class GiftBarrageItemBuilder extends CustomBarrageBuilder {
    @override
    Widget buildWidget(BuildContext context, Barrage barrage) { /// 当
shouldCustomizeBarrageItem 返回true时，自定义Widget
        return const Text(
            barrage.content,
            style: TextStyle(fontSize: 18, fontWeight: FontWeight.w700,
color: Colors.red),
        );
    }

    @override
    bool shouldCustomizeBarrageItem(Barrage barrage) { /// 当弹幕消息的
数据模型，判断当前弹幕消息是否需要自定义
        if (barrage.extInfo.keys.isNotEmpty) {
            return true;
        }
        return false;
    }
}
```



```
}

/// 2、给 BarrageDisplayWidget 设置 setCustomBarrageBuilder
_displayController.setCustomBarrageBuilder(GiftBarrageItemBuilder());
```

## Android

弹幕消息样式默认有两种：普通弹幕消息样式 和 自定义消息样式。具体样式使用整数表示，普通弹幕消息样式是0。

如果您需要更多消息样式（如礼物发送回显），需要重写 BarrageStreamView 的代理 BarrageItemTypeDelegate，并实现新增样式 BarrageItemAdapter。

- 重写 BarrageItemTypeDelegate 代理，支持新的样式 GIFT\_VIEW\_TYPE\_1。

```
public static final int    GIFT_VIEW_TYPE_1        = 1;
public class BarrageViewTypeDelegate implements
BarrageItemTypeDelegate {

    @Override
    public int getItemViewType(int position, Barrage barrage) {
        if (barrage.extInfo != null &&
barrage.extInfo.containsKey(GIFT_VIEW_TYPE)) {
            String viewTypeString =
String.valueOf(barrage.extInfo.get(GIFT_VIEW_TYPE));
            if
(String.valueOf(GIFT_VIEW_TYPE_1).equals(viewTypeString)) {
                return GIFT_VIEW_TYPE_1;
            }
        }
        return 0;
    }
}

mBarrageStreamView.setItemTypeDelegate(new
BarrageViewTypeDelegate());
```

- 实现自定义样式的 adapter，并设置给样式 GIFT\_VIEW\_TYPE\_1。

```
public class GiftBarrageAdapter implements BarrageItemAdapter {
    @Override
```

```
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup
parent) {
    LinearLayout ll = new LinearLayout(mContext);
    ll.addView(new TextView(mContext));
    return new GiftViewHolder(ll, mDefaultGiftIcon);
}

@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder,
int position, Barrage barrage) {
    GiftViewHolder viewHolder = (GiftViewHolder) holder;
    viewHolder.bind(barrage);
}
// GiftViewHolder
...
}

mBarrageStreamView.setItemAdapter(GIFT_VIEW_TYPE_1, new
GiftBarrageAdapter(mContext));
```

## 常见问题

### 弹幕组件初始化时机

由于弹幕组件的 controller 需要接入一些直播间信息参数，所以需要在观众进入直播间或者主播创建直播间后，再加载弹幕组件。

# 互动礼物（TUILiveKit）

最近更新时间：2025-07-23 10:01:22

## 功能介绍

互动礼物组件是一款虚拟礼物互动平台，旨在为用户的社交体验增添更多乐趣。借助这一组件，用户能够向自己欣赏的直播主播送上虚拟礼物，以此展示他们的赞赏、喜爱以及支持。

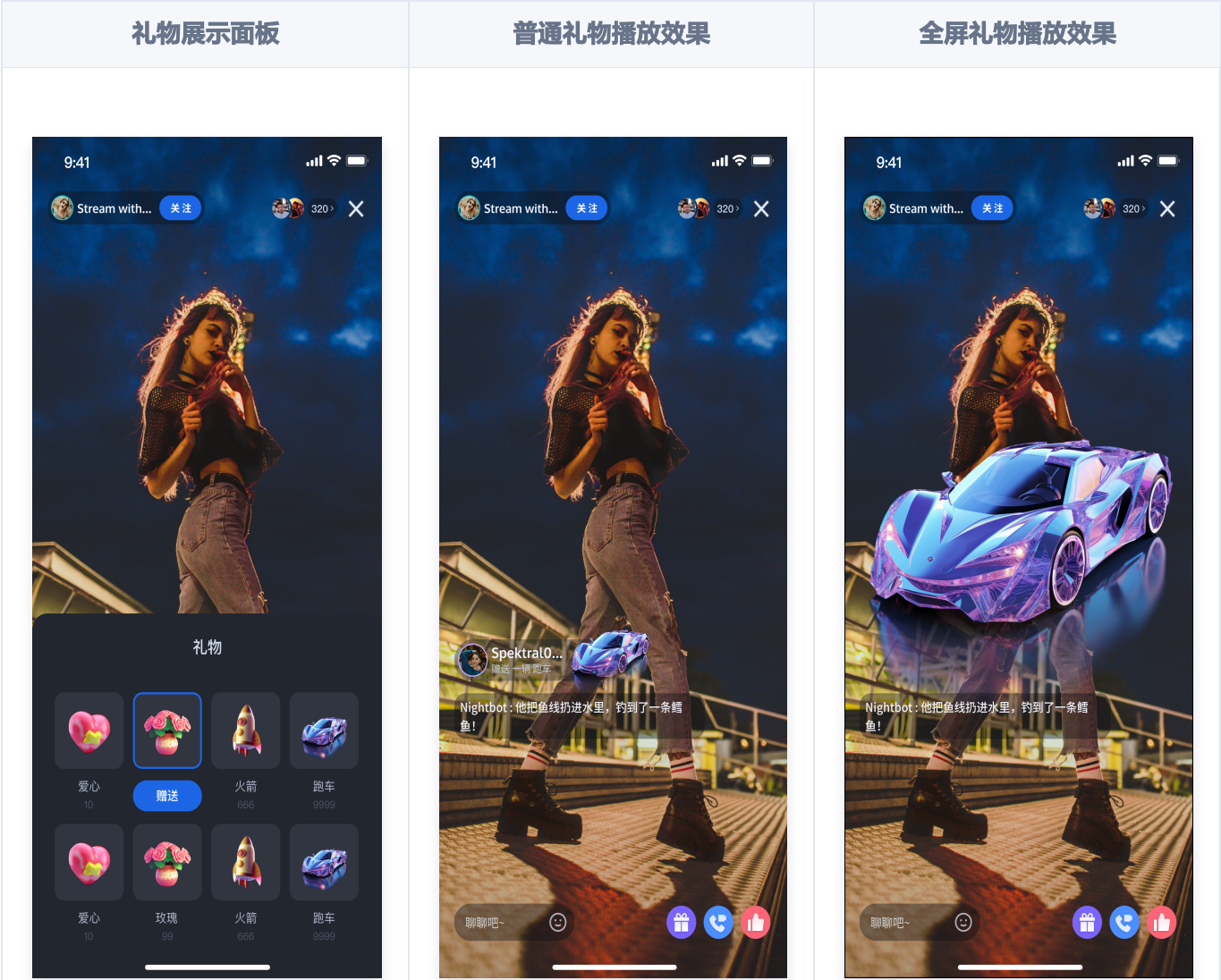
互动礼物组件支持设置点赞、礼物发送面板、发送礼物、播放普通礼物动画、播放 svga 礼物动画等。

### Flutter

互动礼物组件提供 2 个视图组件：

- `GiftSendWidget`：发送礼物消息视图，点击后可以拉起礼物面板。
- `GiftDisplayWidget`：接收礼物消息视图，会显示礼物消息到该视图上。

效果如下图所示：

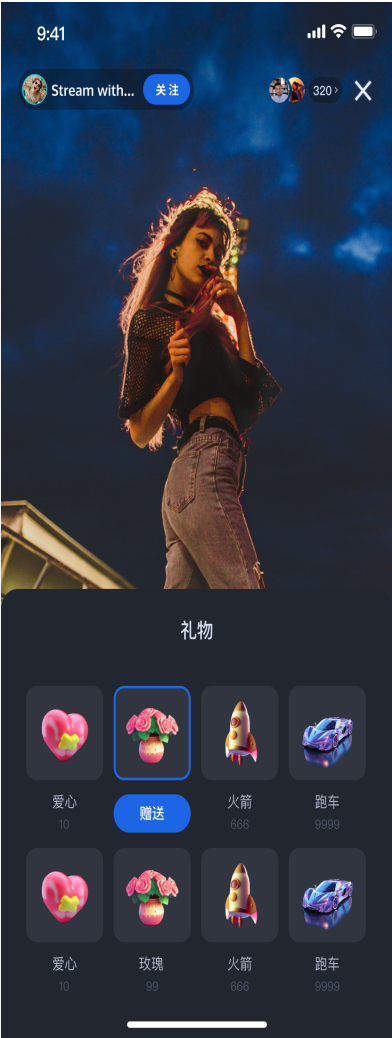


Android

礼物组件主要提供2个 API:

- `GiftListPanelView` : 礼物面板, 呈现礼物列表, 发送礼物及充值。
- `GiftPlayView` : 播放礼物的面板, 自动监听礼物消息。

效果如下图所示:

| 礼物展示面板                                                                             | 普通礼物播放效果                                                                           | 全屏礼物播放效果                                                                             |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|  |  |  |

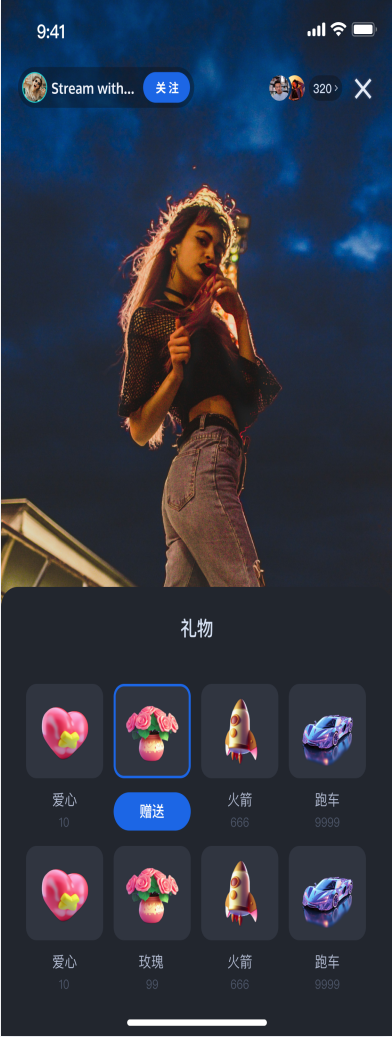


iOS

礼物组件主要提供2个 API：

- GiftListView ：礼物面板，呈现礼物列表，发送礼物。
- GiftPlayView ：播放礼物的面板，自动监听礼物消息。

效果如下图所示：

| 礼物展示面板                                                                             | 普通礼物播放效果                                                                           | 全屏礼物播放效果                                                                             |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
|  |  |  |

快速接入

❗ 说明：

如果您已经集成了 LiveKit 组件，我们在 LiveKit 中默认已经接入了 gift 组件，您可以直接在 LiveKit 中体验互动礼物功能。

## 添加 Gift 组件到项目中

### Flutter

1. 在项目工程的 `pubspec.yaml` 文件的 `dependencies` 节点中，添加 `gift` 的依赖。

```
dependencies:
  flutter:
    sdk: flutter
  flutter_localizations:
    sdk: flutter
  intl: ^0.19.0
  # 添加 gift 依赖
  live_uikit_gift: ^1.0.0
```

2. 执行 `flutter pub get` 命令

3. 配置多语言支持，在您应用 `MaterialApp` 的 `localizationsDelegates` 和 `supportedLocales` 属性上添加 `gift` 组件的多语言支持。

```
MaterialApp(localizationsDelegates: const [
  ...GiftLocalizations.localizationsDelegates,
], supportedLocales: const [
  ...GiftLocalizations.supportedLocales,
],
// ...
);
```

### Android

集成 LiveKit 组件即可正常使用，[详情见 GitHub](#)。

### iOS

集成 LiveKit 组件即可正常使用，[详情见 GitHub](#)。

## 接入发送礼物组件

### Flutter

在您需要接入发送礼物消息的地方构建 `GiftSendController` 和 `GiftSendWidget` 对象，并将构建的 `GiftSendWidget` 对象添加到您的 `Widget` 树上。示例代码如下：

```
GiftSendController _giftSendController = GiftSendController(  
  roomId: "roomId",          /// roomId 替换为您的直播间ID  
  owner: ownerInfo,          /// ownerInfo 替换为您的直播间主播  
  信息  
  self: selfInfo,            /// selfInfo 替换为您的登录用户信  
  息  
);  
  
GiftSendWidget(controller: _giftSendController);
```

### Android

实现 `GiftListPanelView` 的 `OnSendGiftListener` 中的 `onSendGift` 回调，获取礼物个数和礼物信息，在预处理完后可调用 `GiftListPanelView` 的 `sendGift` 函数用于礼物的实际发送。

```
public void onSendGift(GiftListPanelView giftListView, Gift gift, int  
giftCount) {  
  //...此处为预处理，如校验当前用户的余额等操作  
  GiftUser receiver = new GiftUser();  
  //...此处设置礼物接受者信息  
  giftListView.sendGift(gift, giftCount, receiver);  
}
```

### iOS

创建 `GiftListView` 对象，并将其添加到您的视图上，同时还需要实现 `GiftListViewDelegate` 中的 `onSendGift` 回调方法。

当您点击 `GiftListView` 面板上的发送按钮时，您会收到该回调方式，您可在该方法中获取准备发送的礼物个数和礼物信息，在预处理完后可调用 `GiftListView` 的 `sendGift` 函数用于礼物的实际发送。

```
lazy var giftListView = GiftListView(roomId: roomId, delegate: self)
// ...此处将giftListView添加到您的父视图上并调整布局

func onSendGift(gift model: TUIGift, giftCount: Int) {
    //...此处为预处理，如校验当前用户的余额等操作
    let receiver = TUIGiftUser()
    //...此处设置礼物接受者信息
    giftListView.sendGift(model: giftModel, giftCount: giftCount,
        receiver: receiver, completion: completion)
}
```

## 接入显示礼物组件

### Flutter

在您需要接入显示礼物消息的地方构建 `GiftDisplayController` 和 `GiftDisplayWidget` 对象，并将构建的 `GiftDisplayWidget` 对象添加到您的 `Widget` 树上。示例代码如下：

```
GiftDisplayController _giftDisplayController = GiftDisplayController(
    roomId: "roomId",           /// roomId      替换为您的直播间ID
    owner: ownerInfo,           /// ownerInfo 替换为您的直播间主
    播信息
    self: selfInfo,             /// selfInfo   替换为您的登录用户
    信息
);

GiftDisplayWidget(controller: _giftDisplayController!);
```

### Android

礼物展示组件 `GiftPlayView` 自身会接收并播放礼物消息。

```
GiftPlayView giftPlayView = new GiftPlayView(mContext, roomId);
```

### iOS



礼物展示组件 `GiftPlayView` 自身会接收并播放礼物消息。  
您只需要创建 `GiftPlayView` 对象并将其添加到您的视图上即可。

```
let giftPlayView = GiftPlayView(groupId: roomId)
// ...此处将giftPlayView添加到您的父视图上并调整布局
```

## 监听礼物收发消息

### Flutter

若需要获取收发礼物的回调信息，您可以调用 `GiftDisplayController` 的 `setGiftCallback` 方法, 示例代码如下所示：

```
_giftDisplayController?.setGiftCallback(
  onReceiveGiftCallback: _onReceiveGiftCallback,    ///
  _onReceiveGiftCallback 可以替换为您的自定义处理方法
  onSendGiftCallback: _onSendGiftCallback,          ///
  _onSendGiftCallback    可以替换为您的自定义处理方法
);
```

### Android

若需要获取接收礼物的回调信息，可实现 `GiftPlayView` 的 `TUIGiftPlayViewListener` 中的 `onReceiveGift` 函数。

```
public interface TUIGiftPlayViewListener {
    void onReceiveGift (Gift gift, int giftCount, GiftUser
sender, GiftUser receiver);
    //...
}
```

### iOS

若需要获取接收礼物的回调信息，可实现 `GiftPlayView` 的代理 `GiftPlayViewDelegate` 中的 `giftPlayView:onReceiveGift:gift:giftCount:sender:receiver` 函数。

```
func giftPlayView(_ giftPlayView: GiftPlayView, onReceiveGift gift:
    TUIGift, giftCount: Int, sender: TUIGiftUser, receiver: TUIGiftUser)
    // 自定义处理
}
```

## 自定义礼物面板

### Flutter

如果您需要修改礼物面板的礼物列表，您可以调用 `GiftSendController` 的 `setGiftList` 方法，示例如下：

```
List<GiftModel> giftList = [    /// 自定义您的礼物列表数据源
    GiftModel(giftId: "1", giftName: "鸡蛋", imageUrl: "giftImageUrl",
price: 10),
    GiftModel(giftId: "2", giftName: "星星", imageUrl: "giftImageUrl",
price: 10),
];

_giftSendController.setGiftList(giftList);
```

### Android

如果您需要修改礼物面板的礼物列表：

```
mGiftCloudServer.queryGiftInfoList((error, result) -> post(() -> {
    if (error == Error.NO_ERROR) {
        mGiftListView.setGiftList(result);
    } else {
        ToastUtil.toastLongMessage("query gift list error, code = " +
error);
    }
}));
```

**说明：**

- 客户自行实现 `queryGiftInfoList` 方法的逻辑，得到一个自定义的礼物列表 `List< Gift >`，通过 `GiftListView.setGiftList` 设置礼物列表即可。
- 礼物的 `animationUrl` 要求是一个 SVGA 动画。

## iOS

如果您需要修改礼物面板的礼物列表：

```
// 文件位置：
iOS/TUILiveKit/Sources/Component/Gift/Store/TUIGiftStore.swift

class TUIGiftStore {
    static let shared = TUIGiftStore()

    private init() {
        giftCloudServer.queryGiftInfoList { [weak self] error,
giftList in
            guard let self = self else { return }
            if error == .noError {
                self.giftList = giftList
            }
        }
    }

    var giftList: [TUIGift] = []
    let giftCloudServer: IGiftCloudServer = GiftCloudServer()
}

// 文件位置：
iOS/TUILiveKit/Sources/LiveStream/Main/Manager/LSRouterControlCenter.s
wift
private func getRouteDefaultView(route: LSRoute) -> UIView?
{
    switch route {
    case .giftView:
        let giftPanel = GiftListPanel(roomId: manager.roomState.roomId,
dataSource: manager)
        giftPanel.setGiftList(TUIGiftStore.shared.giftList)
        view = giftPanel
    }
}
```

```
}
```

```
}
```

#### ❗ 说明:

- 客户自行实现 `queryGiftInfoList` 方法的逻辑，得到一个自定义的礼物列表 `List<TUIGift>`，通过 `GiftListView.setGiftList` 设置礼物列表即可。
- 礼物的 `animationUrl` 要求是一个SVGA动画。

## 常见问题

### 礼物组件初始化时机

由于礼物组件的 `controller` 需要接入一些直播间信息参数 和 自己的用户信息，所以需要在观众进入直播间或者主播创建直播间后，再加载礼物组件。

# 礼物特效（TUILiveKit）

最近更新时间：2025-04-03 17:08:42

TUILiveKit 提供了两种礼物特效播放器：基础特效播放器和高级特效播放器。默认情况下，集成了基础特效播放器，若您对播放器有更高的性能要求，或期望支持更多的动画文件格式，我们还提供了高级特效播放器供您使用。

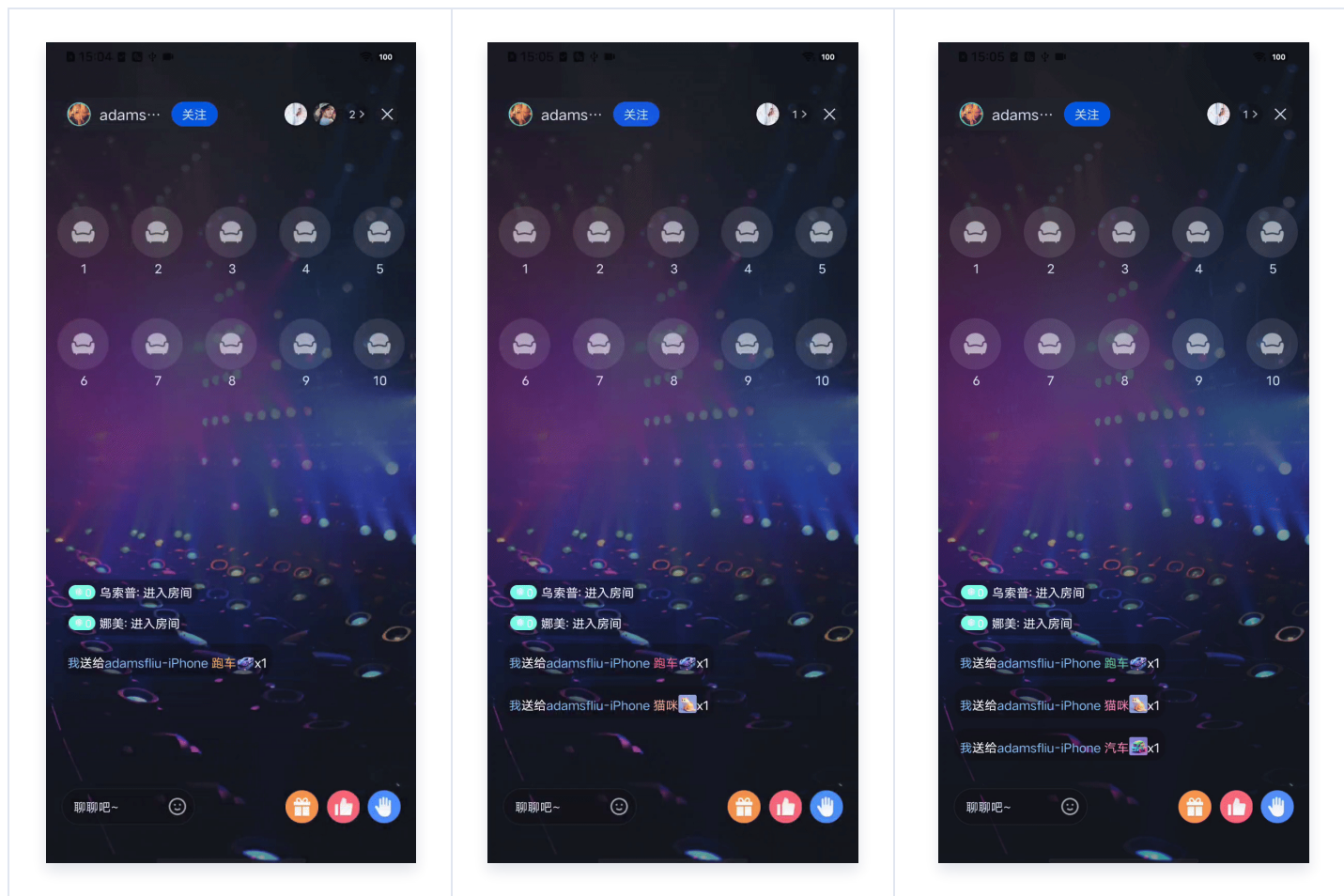
## 基础特效播放器

基础特效播放器基于 `SVGA` 实现，特效动画仅支持 `SVGA` 格式文件。  
使用基础特效播放器时，有如下3个特效动画：



## 效果展示

|    |    |    |
|----|----|----|
| 跑车 | 猫咪 | 汽车 |
|----|----|----|



## 高级特效播放器

TUILiveKit 高级特效播放器采用 [腾讯特效播放器](#)，支持多种格式的特效动画（如：vap、Lottie、mp4、svg 等）。

### 说明：

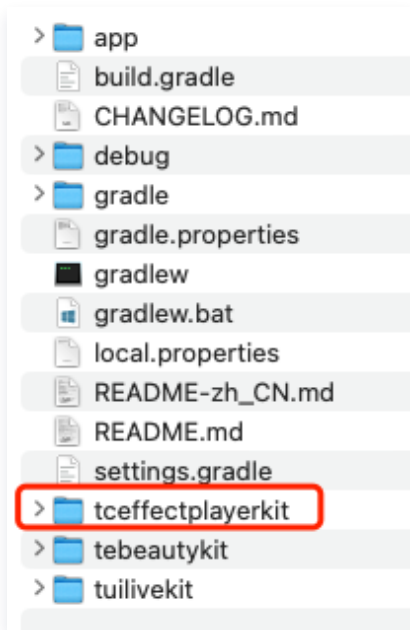
高级特效播放器需单独付费，详情请查看 [腾讯特效播放器](#)。

## 开始集成

### 步骤一：集成礼物特效组件

#### Android

1. 下载并解压 [TUILiveKit](#)，把 `Android/tceffectplayerkit` 文件夹拷贝到自己的工程中，和 `app` 同级目录。



2. 编辑您自己工程的 `settings.gradle` 文件，添加下面的代码：

```
include ':tceffectplayerkit'
```

3. 如果您只接入了高级特效播放器，没有接入高级美颜，那么请在 `tceffectplayerkit/build.gradle` 文件中注释一行代码：

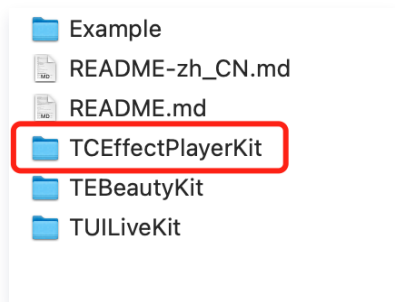
```
dependencies {  
    api("com.tencent.mediacloud:TCEffectPlayer:3.0.0.250") {  
        // exclude group: "com.tencent.mediacloud", module:  
        "TCXMagicAuth"  
    }  
    ...  
}
```

4. 如果您的 App 开启了混淆，请在您的混淆配置里加上如下配置：

```
-keep class com.tcmediax.** { *; }  
-keep class com.tencent.** { *; }  
-keep class com.tencent.xmagic.** { *; }  
-keep class androidx.exifinterface.** {*;}
```

## iOS

1. 下载并解压 **TUILiveKit**，把 `iOS/TCEffectPlayerKit` 文件夹拷贝到自己的工程中，和 `Podfile` 同级目录。



2. 编辑 `podfile` 文件，添加下面的代码：

- 若您未集成 **高级美颜**，参考如下：

```
pod 'TCEffectPlayerKit', :path =>
  '../TCEffectPlayerKit/TCEffectPlayerKit.podspec'
pod 'TCMediaX', :podspec => 'https://mediacloud-
76607.gzc.vod.tencent-
cloud.com/MediaX/iOS/podspec/release/3.0.246/TCMediaX.podspec'
pod 'TCEffectPlayer', :podspec => 'https://mediacloud-
76607.gzc.vod.tencent-
cloud.com/MediaX/iOS/podspec/release/3.0.246/TCEffectPlayer.podspec'
pod 'YTCommonXMagic', :podspec => 'https://mediacloud-
76607.gzc.vod.tencent-
cloud.com/MediaX/iOS/podspec/release/YTCommonXMagic_1.1/YTCommon
XMagic.podspec'
```

- 若您集成了 **高级美颜**，参考如下：

```
pod 'TCEffectPlayerKit/Beauty', :path =>
  '../TCEffectPlayerKit/TCEffectPlayerKit.podspec'
pod 'TCMediaX', :podspec => 'https://mediacloud-
76607.gzc.vod.tencent-
cloud.com/MediaX/iOS/podspec/release/3.0.246/TCMediaX.podspec'
pod 'TCEffectPlayer', :podspec => 'https://mediacloud-
76607.gzc.vod.tencent-
cloud.com/MediaX/iOS/podspec/release/3.0.246/TCEffectPlayer.podspec'
```

3. 终端执行 `pod install` 命令。



## 步骤二：鉴权

1. 申请授权，得到 `LicenseUrl` 和 `LicenseKey` 。详情请发送邮件咨询：[TRTC\\_helper@tencent.com](mailto:TRTC_helper@tencent.com)。
2. 在您的业务初始化的地方，添加如下鉴权代码，并替换您申请的 `LicenseUrl` 和 `LicenseKey` ：  
如：

### Android

```
TCMediaXBase.getInstance().setLicense(context,
    "LicenseUrl", // 请替换 LicenseUrl
    "LicenseKey", // 请替换 LicenseKey
    new ILicenseCallback() {
        @Override
        public void onResult(int error, String message) {
            Log.i("TCMediaXBase", "setLicense result: " + error +
" " + message);
        }
    });
```

### iOS

iOS 可在 `AppDelegate` 的 `didFinishLaunchingWithOptions` 方法中设置相关内容。

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

import TCMediaX

func application(_ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    TCMediaXBase.getInstance().setDelegate(self)
    TCMediaXBase.getInstance().setLicenceURL("LicenseURL", key:
"LicenseKEY")
    return true
}
```

```
func onLicenseCheckCallback(_ errcode: Int32, withParam param:
[AnyHashable : Any]) {
    debugPrint("[TCMediaXBase] setLicense result: errcode:\(errcode),
param:\(param) ")
}
```

# 客户端API（TUILiveKit）

## Android

## VoiceRoomKit

最近更新时间：2025-04-03 17:08:42

### API 简介

VoiceRoomKit 是语音聊天室组件的含UI接口，使用 VoiceRoomKit API，您可以通过简单接口快速实现一个语音聊天室，如果您想要体验并调试语音聊天室效果，请阅读 [Demo 快速跑通](#)。如果您想把我们的功能直接集成到您的项目中，请阅读 [快速接入\(TUILiveKit\)](#)。

#### ❗ 说明：

如果您有自己的 UI 设计，想结合 VoiceRoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看 [RTC Room Engine SDK API](#) 文档，了解更多功能。

### API 概述

| API                            | 描述                      |
|--------------------------------|-------------------------|
| <a href="#">createInstance</a> | 获取一个 VoiceRoomKit 对象实例。 |
| <a href="#">createRoom</a>     | 创建一个语音聊天室直播间。           |
| <a href="#">enterRoom</a>      | 通过 roomId 进入一个语音聊天室直播间。 |

#### ❗ 说明：

如果这组含 UI 的 API 并不能满足您的需求，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 [API 文档](#)，了解更多功能。

### API 详情

#### createInstance

获取一个 VoiceRoomKit 对象实例。

```
static VoiceRoomKit createInstance(Context context)
```

#### 参数：

| 参数      | 类型      | 说明 | 默认值 | 含义            |
|---------|---------|----|-----|---------------|
| context | Context | 必填 | —   | android 上下文对象 |

返回值: VoiceRoomKit

## createRoom

创建一个语音聊天室直播间。

```
void createRoom(String roomId, VoiceRoomDefine.CreateRoomParams params);
```

参数:

| 参数     | 类型               | 说明 | 默认值 | 含义      |
|--------|------------------|----|-----|---------|
| roomId | String           | 必填 | —   | 直播间 ID  |
| params | CreateRoomParams | 必填 | —   | 创建直播间参数 |

返回值: void

## enterRoom

通过 roomId 进入一个语音聊天室直播间。

```
void enterRoom(String roomId);
```

参数:

| 参数     | 类型     | 说明 | 默认值 | 含义     |
|--------|--------|----|-----|--------|
| roomId | String | 必填 | —   | 直播间 ID |

返回值: void

## VoiceRoomDefine 简介

VoiceRoomKit 是语音聊天室的 UIKit 层 数据模型类，主要包含如下数据结构:

### CreateRoomParams

创建语音聊天室直播间时的参数对象，主要包含如下配置参数：

| 参数             | 类型     | 说明 | 默认值 | 含义                                                                                                                                                               |
|----------------|--------|----|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| roomName       | String |    | ""  | 语音聊天室名称                                                                                                                                                          |
| maxAnchorCount | int    |    | 10  | 最大上麦的用户数                                                                                                                                                         |
| seatMode       | String | 必填 |     | 上麦模式，分为如下两种： <ul style="list-style-type: none"><li>• FREE_TO_TAKE(1)：自由上麦模式，台下观众可以自由上麦，无需申请。</li><li>• APPLY_TO_TAKE(2)：申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦。</li></ul> |

# SeatGridView

最近更新时间：2025-04-03 17:08:42

## API 简介

SeatGridView 是我们开发语音聊天室 UIKit 的一个基础控件，该核心控件提供了开启语音聊天室、关闭语音聊天室，直播间内麦位管理，如申请上麦，邀请上麦，移动麦位，踢人下麦等丰富的 API。

## API 概览

| API                                    | 描述                                                                        |
|----------------------------------------|---------------------------------------------------------------------------|
| <a href="#">SeatGridView</a>           | 创建 SeatGridView 对象，支持 <a href="#">代码创建</a> 和 <a href="#">xml 加载</a> 两种方式。 |
| <a href="#">startMicrophone</a>        | 打开本地麦克风                                                                   |
| <a href="#">stopMicrophone</a>         | 关闭本地麦克风                                                                   |
| <a href="#">muteMicrophone</a>         | 暂停发布本地的音频流                                                                |
| <a href="#">unmuteMicrophone</a>       | 恢复发布本地的音频流                                                                |
| <a href="#">startVoiceRoom</a>         | 主播创建直播间并开始推流                                                              |
| <a href="#">stopVoiceRoom</a>          | 主播停止推流并销毁直播间                                                              |
| <a href="#">joinVoiceRoom</a>          | 观众加入某个主播的直播间                                                              |
| <a href="#">leaveVoiceRoom</a>         | 观众离开某个主播的直播间                                                              |
| <a href="#">updateRoomSeatMode</a>     | 更新房间麦位模式                                                                  |
| <a href="#">responseRemoteRequest</a>  | 主播响应上麦申请/观众响应上麦邀请                                                         |
| <a href="#">cancelRequest</a>          | 主播取消上麦邀请/观众取消上麦申请                                                         |
| <a href="#">takeSeat</a>               | 上麦                                                                        |
| <a href="#">moveToSeat</a>             | 移麦                                                                        |
| <a href="#">leaveSeat</a>              | 下麦                                                                        |
| <a href="#">takeUserOnSeatByAdmin</a>  | 主播邀请用户上麦                                                                  |
| <a href="#">kickUserOffSeatByAdmin</a> | 主播踢用户下麦                                                                   |

|                                 |                              |
|---------------------------------|------------------------------|
| <code>n</code>                  |                              |
| <code>lockSeat</code>           | 主播锁定麦位（包括位置锁定、音频状态锁定和视频状态锁定） |
| <code>setLayoutMode</code>      | 主播设置麦位列表的布局模式                |
| <code>setSeatViewAdapter</code> | 为麦位视图设置适配器                   |
| <code>addObserver</code>        | 设置事件回调                       |
| <code>removeObserver</code>     | 移除事件回调                       |

## API 详情

### SeatGridView

创建一个 SeatGridView 对象实例。同时支持**代码创建**和 **xml 加载**两种方式创建。

```
public SeatGridView(Context context)
```

参数：

| 参数      | 类型      | 含义            |
|---------|---------|---------------|
| context | Context | Android 上下文对象 |

返回值：SeatGridView

### startMicrophone

打开本地麦克风。

```
void startMicrophone(ActionCallback callback)
```

参数：

| 参数       | 类型             | 含义    |
|----------|----------------|-------|
| callback | ActionCallback | 操作的回调 |

返回值：void

### stopMicrophone

关闭本地麦克风。

```
void stopMicrophone()
```

返回值: void

## muteMicrophone

暂停发布本地的音频流。

```
void muteMicrophone()
```

返回值: void

## unmuteMicrophone

恢复发布本地的音频流。

```
void unmuteMicrophone(ActionCallback callback)
```

参数:

| 参数       | 类型             | 含义    |
|----------|----------------|-------|
| callback | ActionCallback | 操作的回调 |

返回值: void

## startVoiceRoom

主播创建直播间并开始推流。

```
void startVoiceRoom(RoomInfo roomInfo, GetRoomInfoCallback callback)
```

参数:

| 参数       | 类型             | 含义       |
|----------|----------------|----------|
| roomInfo | RoomInfo       | 创建直播间的信息 |
| callback | ActionCallback | 操作的回调    |

返回值: void



## stopVoiceRoom

主播停止推流并销毁直播间。

```
void stopVoiceRoom(ActionCallback callback)
```

参数：

| 参数       | 类型             | 含义    |
|----------|----------------|-------|
| callback | ActionCallback | 操作的回调 |

返回值：void

## joinVoiceRoom

观众加入某个主播的直播间。

```
void joinVoiceRoom(String roomId, GetRoomInfoCallback callback)
```

参数：

| 参数       | 类型             | 含义     |
|----------|----------------|--------|
| roomId   | String         | 直播间 ID |
| callback | ActionCallback | 操作的回调  |

返回值：void

## leaveVoiceRoom

观众离开某个主播的直播间。

```
void leaveVoiceRoom(ActionCallback callback)
```

参数：

| 参数       | 类型             | 含义    |
|----------|----------------|-------|
| callback | ActionCallback | 操作的回调 |

返回值: void

## updateRoomSeatMode

更新房间麦位模式。

```
void updateRoomSeatMode(SeatMode seatMode, ActionCallback callback)
```

参数:

| 参数       | 类型             | 含义                                                                               |
|----------|----------------|----------------------------------------------------------------------------------|
| seatMode | SeatMode       | FREE_TO_TAKE: 自由上麦模式, 观众可以自由上麦, 无需申请;<br>APPLY_TO_TAKE: 申请上麦模式, 观众上麦需要主播同意后才能上麦。 |
| callback | ActionCallback | 操作的回调。                                                                           |

返回值: void

## responseRemoteRequest

主播响应上麦申请/观众响应上麦邀请。

```
void responseRemoteRequest(String userId, boolean agree, ActionCallback callback)
```

参数:

| 参数       | 类型             | 含义                              |
|----------|----------------|---------------------------------|
| userId   | String         | 响应用户的用户 ID, 如果当前身份是观众, ID 可填空   |
| agree    | boolean        | 是否接受请求, true: 同意请求, false: 拒绝请求 |
| callback | ActionCallback | 操作的回调                           |

返回值: void

## cancelRequest

主播取消上麦邀请/观众取消上麦申请

```
void cancelRequest(String userId, ActionCallback callback)
```

参数：

| 参数       | 类型             | 含义                        |
|----------|----------------|---------------------------|
| userId   | String         | 取消的用户 ID，如果当前身份是观众，ID 可填空 |
| callback | ActionCallback | 操作的回调                     |

返回值：void

## takeSeat

上麦（上麦发言模式下，需要申请）

```
void takeSeat(int index, int timeout, VoiceRoomDefine.RequestCallback callback)
```

参数：

| 参数       | 类型             | 含义                                     |
|----------|----------------|----------------------------------------|
| index    | int            | 上麦的麦位编号                                |
| timeout  | int            | 超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调 |
| callback | ActionCallback | 操作的回调                                  |

返回值：void

## moveToSeat

移麦（已经在麦位上的用户才可以调用此函数）

```
void moveToSeat(int index, ActionCallback callback)
```

参数：

| 参数 | 类型 | 含义 |
|----|----|----|
|----|----|----|

|          |                |             |
|----------|----------------|-------------|
| index    | int            | 需要移动过去的麦位编号 |
| callback | ActionCallback | 操作的回调       |

返回值: void

## leaveSeat

主动下麦

```
void leaveSeat(ActionCallback callback)
```

参数:

| 参数       | 类型             | 含义    |
|----------|----------------|-------|
| callback | ActionCallback | 操作的回调 |

返回值: void

## takeUserOnSeatByAdmin

主播邀请用户上麦

```
void takeUserOnSeatByAdmin(int index, String userId, int timeout, VoiceRoomDefine.RequestCallback callback)
```

参数:

| 参数       | 类型                              | 含义                                         |
|----------|---------------------------------|--------------------------------------------|
| index    | int                             | 被邀请的麦位编号                                   |
| userId   | String                          | 被邀请的用户 ID                                  |
| timeout  | int                             | 超时时间, 单位秒, 如果设置为 0, SDK 不会做超时检测, 也不会触发超时回调 |
| callback | VoiceRoomDefine.RequestCallback | 操作的回调                                      |

返回值: void

## kickUserOffSeatByAdmin

## 主播踢用户下麦

```
void kickUserOffSeatByAdmin(String userId, ActionCallback callback)
```

### 参数：

| 参数       | 类型             | 含义         |
|----------|----------------|------------|
| userId   | String         | 被踢下麦的用户 ID |
| callback | ActionCallback | 操作的回调      |

返回值：void

## lockSeat

锁麦，主播锁定麦位（包括位置锁定、音频状态锁定和视频状态锁定）

```
void lockSeat(int seatIndex, TUIRoomDefine.SeatLockParams params, ActionCallback callback)
```

### 参数：

| 参数        | 类型                           | 含义                                         |
|-----------|------------------------------|--------------------------------------------|
| seatIndex | int                          | 需要锁定麦位的麦位编号                                |
| params    | TUIRoomDefine.SeatLockParams | 锁麦参数。详情参见：<br>TUIRoomDefine.SeatLockParams |
| callback  | ActionCallback               | 操作的回调                                      |

返回值：void

## setLayoutMode

设置麦位列表的布局模式。

```
void setLayoutMode(LayoutMode layoutModel, SeatViewLayoutConfig layoutConfig)
```

### 参数：

| 参数 | 类型 | 含义 |
|----|----|----|
|----|----|----|

|              |                                      |                                   |
|--------------|--------------------------------------|-----------------------------------|
| layoutModel  | <a href="#">LayoutMode</a>           | 麦位列表的布局模式，支持元素布局、宫格布局、纵向布局、自定义布局。 |
| layoutConfig | <a href="#">SeatViewLayoutConfig</a> | 布局配置信息，只有自定义布局模式下才生效              |

返回值：void

## setSeatViewAdapter

为麦位视图设置适配器。

```
void setSeatViewAdapter(VoiceRoomDefine.SeatViewAdapter adapter)
```

参数：

| 参数      | 类型                              | 含义      |
|---------|---------------------------------|---------|
| adapter | <a href="#">SeatViewAdapter</a> | 麦位视图适配器 |

返回值：void

## addObserver

设置事件回调。

```
void addObserver(SeatGridViewObserver observer)
```

参数：

| 参数       | 类型                                   | 含义        |
|----------|--------------------------------------|-----------|
| observer | <a href="#">SeatGridViewObserver</a> | 核心组件的回调对象 |

返回值：void

## removeObserver

移除事件回调。

```
void removeObserver(SeatGridViewObserver observer)
```

参数：

| 参数       | 类型                                   | 含义        |
|----------|--------------------------------------|-----------|
| observer | <a href="#">SeatGridViewObserver</a> | 核心组件的回调对象 |

返回值: void

# SeatGridViewObserver

最近更新时间：2025-04-03 17:08:42

## 事件定义

| 函数列表                                    | 说明                 |
|-----------------------------------------|--------------------|
| <a href="#">onRoomDismissed</a>         | 收到房间销毁的事件          |
| <a href="#">onKickedOutOfRoom</a>       | 收到被移出房间的事件         |
| <a href="#">onSeatRequestReceived</a>   | 收到申请上麦/邀请上麦的请求事件   |
| <a href="#">onSeatRequestCancelled</a>  | 申请麦位/邀请上麦 请求被取消的事件 |
| <a href="#">onKickedOffSeat</a>         | 收到用户被移下麦事件。        |
| <a href="#">onUserAudioStateChanged</a> | 用户音频状态发生变化事件。      |
| <a href="#">onSeatViewClicked</a>       | 麦位视图被点击事件。         |

## 回调事件详情

### onRoomDismissed

直播间被销毁的事件。

```
void onRoomDismissed(String roomId);
```

参数：

| 参数     | 类型     | 描述    |
|--------|--------|-------|
| roomId | String | 房间 ID |

返回值：void

### onKickedOutOfRoom

自己被移出房间的事件。

```
void onKickedOutOfRoom(String roomId, TUIRoomDefine.KickedOutOfRoomReason reason, String message);
```



参数：

| 参数      | 类型                    | 描述     |
|---------|-----------------------|--------|
| roomId  | String                | 房间 ID  |
| reason  | KickedOutOfRoomReason | 被移出原因  |
| message | String                | 被移出的描述 |

返回值：void

## onSeatRequestReceived

收到 申请上麦/邀请上麦 的请求事件。

```
void onSeatRequestReceived(VoiceRoomDefine.RequestType type,
    TUIRoomDefine.UserInfo userInfo);
```

参数：

| 参数       | 类型          | 描述              |
|----------|-------------|-----------------|
| type     | RequestType | 请求类型（申请上麦，邀请上麦） |
| userInfo | UserInfo    | 发送请求的用户信息       |

返回值：void

## onSeatRequestCancelled

申请麦位/邀请上麦 请求被取消的事件。

```
void onSeatRequestCancelled(VoiceRoomDefine.RequestType type,
    TUIRoomDefine.UserInfo userInfo);
```

参数：

| 参数       | 类型          | 描述              |
|----------|-------------|-----------------|
| type     | RequestType | 请求类型（申请上麦，邀请上麦） |
| userInfo | UserInfo    | 取消请求的用户信息       |

返回值：void

## onKickedOffSeat

用户被移下麦事件。

```
void onKickedOffSeat(UserInfo userInfo);
```

参数：

| 参数       | 类型       | 描述           |
|----------|----------|--------------|
| userInfo | UserInfo | 操作移人的主播的用户信息 |

返回值：void

## onUserAudioStateChanged

用户音频状态发生变化事件。

```
void onUserAudioStateChanged(UserInfo userInfo, boolean hasAudio,
    TUIRoomDefine.ChangeReason reason);
```

参数：

| 参数       | 类型           | 描述        |
|----------|--------------|-----------|
| userInfo | UserInfo     | 用户信息      |
| hasAudio | boolean      | 是否有音频流    |
| reason   | ChangeReason | 音频流发生变化原因 |

返回值：void

## onSeatViewClicked

麦位视图被点击事件。

```
void onSeatViewClicked(View seatView, TUIRoomDefine.SeatInfo seatInfo);
```

参数：

| 参数       | 类型   | 描述           |
|----------|------|--------------|
| seatView | View | 当前被点击的麦位视图对象 |

|          |          |      |
|----------|----------|------|
| seatInfo | SeatInfo | 麦位信息 |
|----------|----------|------|

返回值: void

# VoiceRoomDefine

最近更新时间：2025-04-08 14:37:02

## 类型定义

| 类型                         | 描述                               |
|----------------------------|----------------------------------|
| LayoutMode                 | 麦位列表的布局模式，支持元素布局、宫格布局、纵向布局、自定义布局 |
| SeatViewLayoutRowAlignment | 麦位布局的对齐方式                        |
| RequestType                | 请求类型（申请上麦和邀请上麦）                  |
| Size                       | 麦位布局大小                           |
| SeatViewLayoutConfig       | 麦位布局配置信息                         |
| SeatViewLayoutRowConfig    | 麦位布局中每一行的布局配置信息                  |
| RequestCallback            | 请求回调                             |
| SeatViewAdapter            | 麦位视图适配器                          |

## LayoutMode

麦位列表的布局模式

| 类型       | 描述    |
|----------|-------|
| FOCUS    | 元素布局  |
| GRID     | 宫格布局  |
| VERTICAL | 纵向布局  |
| FREE     | 自定义布局 |

## SeatViewLayoutRowAlignment

麦位布局的对齐方式

| 类型    | 说明       |
|-------|----------|
| START | 麦位靠近起始位置 |

|               |                                       |
|---------------|---------------------------------------|
| END           | 麦位靠近结束位置                              |
| CENTER        | 麦位靠近中间位置                              |
| SPACE_BETWEEN | 第一个麦位之前和最后一个麦位之后不留空间，其它麦位之间均匀分布剩余空间   |
| SPACE_AROUND  | 第一个麦位之前和最后一个麦位之后分布一半空间，其它麦位之间均匀分布剩余空间 |
| SPACE_EVENLY  | 在所有麦位之间均匀分布剩余空间                       |

## RequestType

请求类型

| 类型                  | 描述   |
|---------------------|------|
| APPLY_TO_TAKE_SEAT  | 申请上麦 |
| INVITE_TO_TAKE_SEAT | 邀请上麦 |

## Size

麦位布局大小

| 类型     | 描述   |
|--------|------|
| width  | 布局宽度 |
| height | 布局高度 |

## SeatViewLayoutConfig

麦位布局配置信息

| 类型         | 描述                                                           |
|------------|--------------------------------------------------------------|
| rowConfigs | 麦位布局中所有行配置信息列表,内容可参见 <a href="#">SeatViewLayoutRowConfig</a> |
| rowSpacing | 麦位布局行间距                                                      |

## SeatViewLayoutRowConfig

麦位布局中每一行的布局配置信息

| 类型          | 描述                                                       |
|-------------|----------------------------------------------------------|
| count       | 此行显示的麦位数量                                                |
| seatSpacing | 此行中每个麦位水平间距（仅当对齐方式为 START、END 和 CENTER 时生效）              |
| seatSize    | 此行中麦位布局大小                                                |
| alignment   | 此行中布局的对齐方式（ <a href="#">SeatViewLayoutRowAlignment</a> ） |

## RequestCallback

申请上麦/邀请上麦回调

| API                         | 说明    |
|-----------------------------|-------|
| <a href="#">onAccepted</a>  | 请求被接受 |
| <a href="#">onRejected</a>  | 请求被拒绝 |
| <a href="#">onCancelled</a> | 请求被取消 |
| <a href="#">onTimeout</a>   | 请求超时  |
| <a href="#">onError</a>     | 请求异常  |

## SeatViewAdapter

麦位视图适配器接口，您可以通过实现该接口来自定义每一个麦位的显示 UI。

| API                              | 说明           |
|----------------------------------|--------------|
| <a href="#">createSeatView</a>   | 创建单个麦位布局时回调。 |
| <a href="#">updateSeatView</a>   | 更新麦位视图时回调。   |
| <a href="#">updateUserVolume</a> | 更新用户音量时回调。   |

## 回调事件详情

### onAccepted

申请上麦/邀请上麦 请求被接受。

```
void onAccepted(TUIRoomDefine.UserInfo userInfo);
```

参数：

| 参数       | 类型       | 描述          |
|----------|----------|-------------|
| userInfo | UserInfo | 响应当前请求的用户信息 |

返回值：void

## onRejected

申请上麦/邀请上麦 请求被拒绝。

```
void onRejected(TUIRoomDefine.UserInfo userInfo);
```

参数：

| 参数       | 类型       | 描述          |
|----------|----------|-------------|
| userInfo | UserInfo | 响应当前请求的用户信息 |

返回值：void

## onCancelled

申请上麦/邀请上麦请求被取消。

```
void onCancelled(TUIRoomDefine.UserInfo userInfo);
```

参数：

| 参数       | 类型       | 描述          |
|----------|----------|-------------|
| userInfo | UserInfo | 取消当前请求的用户信息 |

返回值：void

## onTimeout

申请上麦/邀请上麦请求超时。

```
void onTimeout(TUIRoomDefine.UserInfo userInfo);
```

参数：

| 参数       | 类型       | 描述        |
|----------|----------|-----------|
| userInfo | UserInfo | 发起请求的用户信息 |

返回值: void

## onError

申请上麦/邀请上麦请求错误。

```
void onError(TUIRoomDefine.UserInfo userInfo, TUICommonDefine.Error error, String message);
```

参数:

| 参数       | 类型                    | 描述        |
|----------|-----------------------|-----------|
| userInfo | UserInfo              | 发起请求的用户信息 |
| error    | TUICommonDefine.Error | 错误码       |
| message  | String                | 错误信息      |

返回值: void

## createSeatView

创建单个麦位布局时回调，您需要将您自定义的 view 作为返回值，核心 view 会帮您创建好视图。

```
View createSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo);
```

参数:

| 参数           | 类型           | 描述      |
|--------------|--------------|---------|
| seatGridView | SeatGridView | 语聊房核心组件 |
| seatInfo     | SeatInfo     | 麦位信息    |

返回值: View

## updateSeatView



更新麦位视图时回调，您可以根据回调回来的 seatInfo 信息来更新自己的麦位视图。

```
void updateSeatView(SeatGridView seatGridView, TUIRoomDefine.SeatInfo seatInfo, View seatView);
```

参数：

| 参数               | 类型           | 描述        |
|------------------|--------------|-----------|
| seatGridVie<br>w | SeatGridView | 语聊房核心组件   |
| seatInfo         | SeatInfo     | 麦位信息      |
| seatView         | View         | 当前更新的麦位视图 |

返回值：void

## updateUserVolume

更新用户音量时回调，您可以根据回调回来的音量更新自己的麦位视图。

```
void updateUserVolume(SeatGridView seatGridView, int volume, View seatView);
```

参数：

| 参数               | 类型           | 描述            |
|------------------|--------------|---------------|
| seatGridVie<br>w | SeatGridView | 语聊房核心组件       |
| volume           | int          | 音量大小          |
| seatView         | View         | 当前音量变化的麦位布局视图 |

返回值：void

# iOS

## VoiceRoomKit

最近更新时间：2025-04-03 17:08:42

### API 简介

VoiceRoomKit 是语音聊天室组件的含UI接口，使用 VoiceRoomKit API，您可以通过简单接口快速实现一个语音聊天室，如果您想要体验并调试语音聊天室效果，请阅读 [Demo 快速跑通](#)。如果您想把我们的功能直接集成到您的项目中，请阅读 [快速接入\(TUILiveKit\)](#)。

#### ⓘ 说明：

如果您有自己的 UI 设计，想结合 VoiceRoomKit 业务自行开发，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看 [RTC Room Engine SDK API](#) 文档，了解更多功能。

### API 概述

| API                            | 描述                      |
|--------------------------------|-------------------------|
| <a href="#">createInstance</a> | 获取一个 VoiceRoomKit 对象实例。 |
| <a href="#">createRoom</a>     | 创建一个语音聊天室直播间。           |
| <a href="#">enterRoom</a>      | 通过 roomId 进入一个语音聊天室直播间。 |

#### ⓘ 说明：

如果这组合 UI 的 API 并不能满足您的需求，我们同时提供有灵活性更好，功能更全面的 RTC Room Engine SDK，您可以通过查看我们的 [API 文档](#)，了解更多功能。

### API 详情

#### createInstance

获取一个 VoiceRoomKit 对象实例。

```
static func createInstance() -> VoiceRoomKit
```

返回值：VoiceRoomKit

#### createRoom

创建一个语音聊天室直播间。

```
func createRoom(roomId: String, params: CreateRoomParams)
```

参数：

| 参数     | 类型               | 说明 | 默认值 | 含义      |
|--------|------------------|----|-----|---------|
| roomId | String           | 必填 | —   | 直播间 ID  |
| params | CreateRoomParams | 必填 | —   | 创建直播间参数 |

返回值：void

## enterRoom

通过 roomId 进入一个语音聊天室直播间。

```
func enterRoom(roomId: String)
```

参数：

| 参数     | 类型     | 说明 | 默认值 | 含义     |
|--------|--------|----|-----|--------|
| roomId | String | 必填 | —   | 直播间 ID |

返回值：void

## VoiceRoomDefine 简介

VoiceRoomKit 是语音聊天室的 UIKit 层 数据模型类，主要包含如下数据结构：

### CreateRoomParams

创建语音聊天室直播间时的参数对象，主要包含如下配置参数：

| 参数 | 类型 | 说明 | 默认值 | 含义 |
|----|----|----|-----|----|
|----|----|----|-----|----|

|                |                             |    |    |                                                                                                                                                                                         |
|----------------|-----------------------------|----|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| roomName       | String                      |    | "" | 语音聊天室名称                                                                                                                                                                                 |
| maxAnchorCount | Int                         |    | 10 | 最大上麦的用户数                                                                                                                                                                                |
| seatMode       | <a href="#">TUISeatMode</a> | 必填 |    | <div>上麦模式，分为如下两种：</div> <ul style="list-style-type: none"><li>• TUISeatModeFreeToTake：自由上麦模式，台下观众可以自由上麦，无需申请。</li><li>• TUISeatModeApplyToTake：申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦。</li></ul> |

# SeatGridView

最近更新时间：2025-04-03 17:08:42

## API 简介

SeatGridView 是我们开发语音聊天室 UIKit 的一个基础控件，该核心控件提供了开启语音聊天室、关闭语音聊天室，直播间内麦位管理，如申请上麦，邀请上麦，移动麦位，踢人下麦等丰富的 API。

## API 概览

| API                                    | 描述                 |
|----------------------------------------|--------------------|
| <a href="#">SeatGridView</a>           | 创建 SeatGridView 对象 |
| <a href="#">startMicrophone</a>        | 打开本地麦克风            |
| <a href="#">stopMicrophone</a>         | 关闭本地麦克风            |
| <a href="#">muteMicrophone</a>         | 暂停发布本地的音频流         |
| <a href="#">unmuteMicrophone</a>       | 恢复发布本地的音频流         |
| <a href="#">startVoiceRoom</a>         | 主播创建直播间并开始推流       |
| <a href="#">stopVoiceRoom</a>          | 主播停止推流并销毁直播间       |
| <a href="#">joinVoiceRoom</a>          | 观众加入某个主播的直播间       |
| <a href="#">leaveVoiceRoom</a>         | 观众离开某个主播的直播间       |
| <a href="#">updateRoomSeatMode</a>     | 更新房间麦位模式           |
| <a href="#">responseRemoteRequest</a>  | 主播响应上麦申请/观众响应上麦邀请  |
| <a href="#">cancelRequest</a>          | 主播取消上麦邀请/观众取消上麦申请  |
| <a href="#">takeSeat</a>               | 上麦                 |
| <a href="#">moveToSeat</a>             | 移麦                 |
| <a href="#">leaveSeat</a>              | 下麦                 |
| <a href="#">takeUserOnSeatByAdmin</a>  | 主播邀请用户上麦           |
| <a href="#">kickUserOffSeatByAdmin</a> | 主播踢用户下麦            |

|                     |                              |
|---------------------|------------------------------|
| n                   |                              |
| lockSeat            | 主播锁定麦位（包括位置锁定、音频状态锁定和视频状态锁定） |
| setLayoutMode       | 主播设置麦位列表的布局模式                |
| setSeatViewDelegate | 为麦位视图设置代理                    |
| addObserver         | 设置事件回调                       |
| removeObserver      | 移除事件回调                       |

## API 详情

### SeatGridView

创建一个 SeatGridView 对象实例。

```
SeatGridView()
```

### startMicrophone

打开本地麦克风。

```
func startMicrophone(onSuccess: @escaping SGOnSuccess,
                    onError: @escaping SGOnError)
```

参数：

| 参数        | 类型          | 含义   |
|-----------|-------------|------|
| onSuccess | SGOnSuccess | 成功回调 |
| onError   | SGOnError   | 失败回调 |

### stopMicrophone

关闭本地麦克风。

```
func stopMicrophone()
```

### muteMicrophone

暂停发布本地的音频流。

```
func muteMicrophone()
```

## unmuteMicrophone

恢复发布本地的音频流。

```
func unmuteMicrophone(onSuccess: @escaping SGO onSuccess,
                      onError: @escaping SGO onError)
```

参数：

| 参数        | 类型            | 含义   |
|-----------|---------------|------|
| onSuccess | SGO onSuccess | 成功回调 |
| onError   | SGO onError   | 失败回调 |

## startVoiceRoom

主播创建直播间并开始推流。

```
func startVoiceRoom(roomInfo: TUIRoomInfo,
                    onSuccess: @escaping SGO onRoomSuccess,
                    onError: @escaping SGO onError)
```

参数：

| 参数        | 类型                | 含义              |
|-----------|-------------------|-----------------|
| roomInfo  | TUIRoomInfo       | 创建直播间的信息        |
| onSuccess | SGO onRoomSuccess | 成功回调（含有新的直播间信息） |
| onError   | SGO onError       | 失败回调            |

## stopVoiceRoom

主播停止推流并销毁直播间。

```
func stopVoiceRoom(onSuccess: @escaping SGO onSuccess,
                   onError: @escaping SGO onError)
```

参数：

| 参数        | 类型            | 含义   |
|-----------|---------------|------|
| onSuccess | SGO onSuccess | 成功回调 |
| onError   | SGO onError   | 失败回调 |

## joinVoiceRoom

观众加入某个主播的直播间。

```
func joinVoiceRoom(roomId: String,
                   onSuccess: @escaping SGO onRoomSuccess,
                   onError: @escaping SGO onError)
```

参数：

| 参数        | 类型                | 含义            |
|-----------|-------------------|---------------|
| roomId    | String            | 直播间 ID        |
| onSuccess | SGO onRoomSuccess | 成功回调（含有直播间信息） |
| onError   | SGO onError       | 失败回调          |

## leaveVoiceRoom

观众离开某个主播的直播间。

```
func leaveVoiceRoom(onSuccess: @escaping SGO onSuccess,
                    onError: @escaping SGO onError)
```

参数：

| 参数 | 类型 | 含义 |
|----|----|----|
|----|----|----|



|           |             |      |
|-----------|-------------|------|
| onSuccess | SGOnSuccess | 成功回调 |
| onError   | SGOnError   | 失败回调 |

## updateRoomSeatMode

更新房间麦位模式。

```
func updateRoomSeatMode(seatMode: TUISeatMode,
                        onSuccess: @escaping SGOnSuccess,
                        onError: @escaping SGOnError)
```

参数：

| 参数        | 类型          | 含义                                                                        |
|-----------|-------------|---------------------------------------------------------------------------|
| seatMode  | TUISeatMode | freeToTake: 自由上麦模式，观众可以自由上麦，无需申请；<br>applyToTake: 申请上麦模式，观众上麦需要主播同意后才能上麦。 |
| onSuccess | SGOnSuccess | 成功回调                                                                      |
| onError   | SGOnError   | 失败回调                                                                      |

## responseRemoteRequest

主播响应上麦申请/观众响应上麦邀请。

```
func responseRemoteRequest(userId: String,
                          agree: Bool,
                          onSuccess: @escaping SGOnSuccess,
                          onError: @escaping SGOnError)
```

参数：

| 参数     | 类型     | 含义                            |
|--------|--------|-------------------------------|
| userId | String | 响应用户的用户 ID，如果当前身份是观众，ID 可填空   |
| agree  | Bool   | 是否接受请求，true: 同意请求，false: 拒绝请求 |

|           |            |      |
|-----------|------------|------|
| onSuccess | SOnSuccess | 成功回调 |
| onError   | SOnError   | 失败回调 |

## cancelRequest

主播取消上麦邀请/观众取消上麦申请

```
func cancelRequest(userId: String,
    onSuccess: @escaping SOnSuccess,
    onError: @escaping SOnError)
```

参数:

| 参数        | 类型         | 含义                        |
|-----------|------------|---------------------------|
| userId    | String     | 取消的用户 ID，如果当前身份是观众，ID 可填空 |
| onSuccess | SOnSuccess | 成功回调                      |
| onError   | SOnError   | 失败回调                      |

## takeSeat

上麦（上麦发言模式下，需要申请）

```
func takeSeat(index: Int,
    timeout: Int,
    onAccepted: @escaping SOnRequestAccepted,
    onRejected: @escaping SOnRequestRejected,
    onCancelled: @escaping SOnRequestCancelled,
    onTimeout: @escaping SOnRequestTimeout,
    onError: @escaping SOnRequestError)
```

参数:

| 参数    | 类型  | 含义      |
|-------|-----|---------|
| index | Int | 上麦的麦位编号 |

|             |                      |                                        |
|-------------|----------------------|----------------------------------------|
| timeout     | Int                  | 超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调 |
| onAccepted  | SGOnRequestAccepted  | 上麦申请被同意回调                              |
| onRejected  | SGOnRequestRejected  | 上麦申请被拒绝回调                              |
| onCancelled | SGOnRequestCancelled | 上麦申请已取消回调                              |
| onTimeout   | SGOnRequestTimeout   | 上麦申请超时回调                               |
| onError     | SGOnRequestError     | 上麦申请错误回调                               |

## moveToSeat

移麦（已经在麦位上的用户才可以调用此函数）

```
func moveToSeat(index: Int,
                onSuccess: @escaping SGOnSuccess,
                onError: @escaping SGOnError)
```

参数：

| 参数        | 类型          | 含义          |
|-----------|-------------|-------------|
| index     | Int         | 需要移动过去的麦位编号 |
| onSuccess | SGOnSuccess | 成功回调        |
| onError   | SGOnError   | 失败回调        |

## leaveSeat

主动下麦

```
func leaveSeat(onSuccess: @escaping SGOnSuccess, onError: @escaping SGOnError)
```

参数:

| 参数        | 类型          | 含义   |
|-----------|-------------|------|
| onSuccess | SGOnSuccess | 成功回调 |
| onError   | SGOnError   | 失败回调 |

## takeUserOnSeatByAdmin

主播邀请用户上麦

```
func takeUserOnSeatByAdmin(index: Int,
                             userId: String,
                             onAccepted: @escaping SGOnRequestAccepted,
                             onRejected: @escaping SGOnRequestRejected,
                             onCancelled: @escaping SGOnRequestCancelled,
                             onTimeout: @escaping SGOnRequestTimeout,
                             onError: @escaping SGOnRequestError)
```

参数:

| 参数          | 类型                   | 含义                                     |
|-------------|----------------------|----------------------------------------|
| index       | Int                  | 被邀请的麦位编号                               |
| userId      | String               | 被邀请的用户 ID                              |
| timeout     | Int                  | 超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调 |
| onAccepted  | SGOnRequestAccepted  | 邀请上麦请求被同意回调                            |
| onRejected  | SGOnRequestRejected  | 邀请上麦请求被拒绝回调                            |
| onCancelled | SGOnRequestCancelled | 邀请上麦请求被取消回调                            |
| onTimeout   | SGOnRequestTimeout   | 邀请上麦请求已超时回调                            |
| onError     | SGOnRequestError     | 邀请上麦请求错误回调                             |

or

## kickUserOffSeatByAdmin

主播踢用户下麦

```
func kickUserOffSeatByAdmin(userId: String,
                             onSuccess: @escaping SGO onSuccess,
                             onError: @escaping SGO onError)
```

参数:

| 参数        | 类型          | 含义         |
|-----------|-------------|------------|
| userId    | String      | 被踢下麦的用户 ID |
| onSuccess | SGOnSuccess | 成功回调       |
| onError   | SGOnError   | 失败回调       |

## lockSeat

锁麦，主播锁定麦位（包括位置锁定、音频状态锁定和视频状态锁定）

```
func lockSeat(index: Int,
               lockMode: TUISeatLockParams,
               onSuccess: @escaping SGO onSuccess,
               onError: @escaping SGO onError)
```

参数:

| 参数        | 类型                | 含义                                     |
|-----------|-------------------|----------------------------------------|
| seatIndex | int               | 需要锁定麦位的麦位编号                            |
| params    | TUISeatLockParams | 锁麦参数。详情参见:TUIRoomDefine.SeatLockParams |
| onSuccess | SGOnSuccess       | 成功回调                                   |
| onError   | SGOnError         | 失败回调                                   |

## setLayoutMode

设置麦位列表的布局模式。

```
func setLayoutMode(layoutMode: SGLayoutMode,
    layoutConfig: SeatViewLayoutConfig? = nil)
```

参数：

| 参数           | 类型                                     | 含义                               |
|--------------|----------------------------------------|----------------------------------|
| layoutModel  | <a href="#">SGLayoutMode</a>           | 麦位列表的布局模式，支持元素布局、宫格布局、纵向布局、自由布局。 |
| layoutConfig | <a href="#">SGSeatViewLayoutConfig</a> | 布局配置信息，只有自由布局模式下才生效              |

## setSeatViewDelegate

为麦位视图设置适配器。

```
func setSeatViewDelegate(_ delegate: SGSeatViewDelegate)
```

参数：

| 参数       | 类型                                 | 含义     |
|----------|------------------------------------|--------|
| delegate | <a href="#">SGSeatViewDelegate</a> | 麦位视图代理 |

## addObserver

设置事件回调。

```
func addObserver(observer: SeatGridViewObserver)
```

参数：

| 参数       | 类型                                   | 含义        |
|----------|--------------------------------------|-----------|
| observer | <a href="#">SeatGridViewObserver</a> | 核心组件的回调对象 |

## removeObserver

移除事件回调。

```
func removeObserver(observer: SeatGridViewObserver)
```

参数：

| 参数       | 类型                                   | 含义        |
|----------|--------------------------------------|-----------|
| observer | <a href="#">SeatGridViewObserver</a> | 核心组件的回调对象 |

# SeatGridViewObserver

最近更新时间：2025-04-03 17:08:42

## 事件定义

| 函数列表                                 | 说明                |
|--------------------------------------|-------------------|
| <code>onRoomDismissed</code>         | 收到房间销毁的事件         |
| <code>onKickedOutOfRoom</code>       | 收到被移出房间的事件        |
| <code>onSeatRequestReceived</code>   | 收到申请上麦/邀请上麦的请求事件  |
| <code>onSeatRequestCancelled</code>  | 申请麦位/邀请上麦请求被取消的事件 |
| <code>onKickedOffSeat</code>         | 收到用户被移下麦事件。       |
| <code>onUserAudioStateChanged</code> | 用户音频状态发生变化事件。     |
| <code>onSeatViewClicked</code>       | 麦位视图被点击事件。        |

## 回调事件详情

### onRoomDismissed

直播间被销毁的事件。

```
func onRoomDismissed(roomId: String)
```

参数：

| 参数     | 类型     | 描述    |
|--------|--------|-------|
| roomId | String | 房间 ID |

### onKickedOutOfRoom

自己被移出房间的事件。

```
func onKickedOutOfRoom(roomId: String,
                        reason: TUIKickedOutOfRoomReason,
                        message: String)
```

参数：



| 参数      | 类型                       | 描述     |
|---------|--------------------------|--------|
| roomId  | String                   | 房间 ID  |
| reason  | TUIKickedOutOfRoomReason | 被移出原因  |
| message | String                   | 被移出的描述 |

## onSeatRequestReceived

收到申请上麦/邀请上麦的请求事件。

```
func onSeatRequestReceived(type: SRequestType,
                           userInfo: TUIUserInfo)
```

参数：

| 参数       | 类型                           | 描述              |
|----------|------------------------------|-----------------|
| type     | <a href="#">SRequestType</a> | 请求类型（申请上麦，邀请上麦） |
| userInfo | TUIUserInfo                  | 发送请求的用户信息       |

## onSeatRequestCancelled

申请麦位/邀请上麦请求被取消的事件。

```
func onSeatRequestCancelled(type: SRequestType,
                             userInfo: TUIUserInfo)
```

参数：

| 参数       | 类型                           | 描述              |
|----------|------------------------------|-----------------|
| type     | <a href="#">SRequestType</a> | 请求类型（申请上麦，邀请上麦） |
| userInfo | TUIUserInfo                  | 取消请求的用户信息       |

## onKickedOffSeat

用户被移下麦事件。

```
func onKickedOffSeat (userInfo: TUIUserInfo)
```

参数：

| 参数       | 类型          | 描述           |
|----------|-------------|--------------|
| userInfo | TUIUserInfo | 操作移人的主播的用户信息 |

## onUserAudioStateChanged

用户音频状态发生变化事件。

```
func onUserAudioStateChanged (userInfo: TUIUserInfo,  
                               hasAudio: Bool,  
                               reason: TUICChangeReason)
```

参数：

| 参数       | 类型               | 描述        |
|----------|------------------|-----------|
| userInfo | TUIUserInfo      | 用户信息      |
| hasAudio | Bool             | 是否有音频流    |
| reason   | TUICChangeReason | 音频流发生变化原因 |

## onSeatViewClicked

麦位视图被点击事件。

```
func onSeatViewClicked (seatView: UIView, seatInfo: TUISeatInfo)
```

参数：

| 参数       | 类型          | 描述           |
|----------|-------------|--------------|
| seatView | View        | 当前被点击的麦位视图对象 |
| seatInfo | TUISeatInfo | 麦位信息         |

# VoiceRoomDefine

最近更新时间：2025-04-03 17:08:43

## 类型定义

| 类型                                           | 描述                               |
|----------------------------------------------|----------------------------------|
| <a href="#">SGLayoutMode</a>                 | 麦位列表的布局模式，支持元素布局、宫格布局、纵向布局、自定义布局 |
| <a href="#">SGSeatViewLayoutRowAlignment</a> | 麦位布局的对齐方式                        |
| <a href="#">SGRequestType</a>                | 请求类型（申请上麦和邀请上麦）                  |
| <a href="#">SGSeatViewLayoutConfig</a>       | 麦位布局配置信息                         |
| <a href="#">SGSeatViewLayoutRowConfig</a>    | 麦位布局中每一行的布局配置信息                  |
| <a href="#">SGSeatViewDelegate</a>           | 麦位视图代理                           |

## SGLayoutMode

麦位列表的布局模式

| 类型       | 描述    |
|----------|-------|
| focus    | 元素布局  |
| grid     | 宫格布局  |
| vertical | 纵向布局  |
| free     | 自定义布局 |

## SGSeatViewLayoutRowAlignment

麦位布局的对齐方式

| 类型     | 说明       |
|--------|----------|
| start  | 麦位靠近起始位置 |
| end    | 麦位靠近结束位置 |
| center | 麦位靠近中间位置 |

|              |                                       |
|--------------|---------------------------------------|
| spaceBetween | 第一个麦位之前和最后一个麦位之后不留空间，其它麦位之间均匀分布剩余空间   |
| spaceAround  | 第一个麦位之前和最后一个麦位之后分布一半空间，其它麦位之间均匀分布剩余空间 |
| spaceEvenly  | 在所有麦位之间均匀分布剩余空间                       |

## SGRequestType

请求类型

| 类型               | 描述   |
|------------------|------|
| applyToTakeSeat  | 申请上麦 |
| inviteToTakeSeat | 邀请上麦 |

## SGSeatViewLayoutConfig

麦位布局配置信息

| 类型         | 描述                                                             |
|------------|----------------------------------------------------------------|
| rowConfigs | 麦位布局中所有行配置信息列表，内容可参见 <a href="#">SGSeatViewLayoutRowConfig</a> |
| rowSpacing | 麦位布局行间距                                                        |

## SGSeatViewLayoutRowConfig

麦位布局中每一行的布局配置信息

| 类型          | 描述                                                         |
|-------------|------------------------------------------------------------|
| count       | 此行显示的麦位数量                                                  |
| seatSpacing | 此行中每个麦位水平间距（仅当对齐方式为 START、END 和 CENTER 时生效）                |
| seatSize    | 此行中麦位布局大小                                                  |
| alignment   | 此行中布局的对齐方式（ <a href="#">SGSeatViewLayoutRowAlignment</a> ） |

## SGSeatViewDelegate

麦位视图适配器接口，您可以通过实现该接口来自定义每一个麦位的显示UI。

| API                                           | 说明           |
|-----------------------------------------------|--------------|
| <a href="#">seatGridView:createSeatView</a>   | 创建单个麦位布局时回调。 |
| <a href="#">seatGridView:updateSeatView</a>   | 更新麦位视图时回调。   |
| <a href="#">seatGridView:updateUserVolume</a> | 更新用户音量时回调。   |

## 回调事件详情

### onAccepted

申请上麦/邀请上麦 请求被接受。

```
onAccepted: (_ userInfo: TUIUserInfo) -> Void
```

参数：

| 参数       | 类型          | 描述          |
|----------|-------------|-------------|
| userInfo | TUIUserInfo | 响应当前请求的用户信息 |

### onRejected

申请上麦/邀请上麦 请求被拒绝。

```
onRejected: (_ userInfo: TUIUserInfo) -> Void
```

参数：

| 参数       | 类型          | 描述          |
|----------|-------------|-------------|
| userInfo | TUIUserInfo | 响应当前请求的用户信息 |

### onCancelled

申请上麦/邀请上麦 请求被取消。

```
onCancelled: (_ userInfo: TUIUserInfo) -> Void
```

参数：

| 参数       | 类型          | 描述          |
|----------|-------------|-------------|
| userInfo | TUIUserInfo | 取消当前请求的用户信息 |

## onTimeout

申请上麦/邀请上麦 请求超时。

```
onTimeout: (_ userInfo: TUIUserInfo) -> Void
```

参数：

| 参数       | 类型       | 描述        |
|----------|----------|-----------|
| userInfo | UserInfo | 发起请求的用户信息 |

## onError

申请上麦/邀请上麦 请求错误。

```
onError: (_ userInfo: TUIUserInfo, _ code: Int, _ message: String) -> Void
```

参数：

| 参数       | 类型                    | 描述        |
|----------|-----------------------|-----------|
| userInfo | TUIUserInfo           | 发起请求的用户信息 |
| error    | TUICommonDefine.Error | 错误码       |
| message  | String                | 错误信息      |

## seatGridView:createSeatView

创建单个麦位布局时回调，您需要将您自定义的view作为返回值，核心view会帮您创建好视图。

```
func seatGridView(_ view: SeatGridView, createSeatView seatInfo: TUISeatInfo) -> UIView?
```

参数：

| 参数 | 类型 | 描述 |
|----|----|----|
|----|----|----|

|              |              |         |
|--------------|--------------|---------|
| seatGridView | SeatGridView | 语聊房核心组件 |
| seatInfo     | SeatInfo     | 麦位信息    |

返回值: UIView? (返回值为nil时使用默认麦位视图)

## seatGridView:updateSeatView

更新麦位视图时回调，您可以根据回调回来的seatInfo信息来更新自己的麦位视图。

```
func seatGridView(_ view: SeatGridView, updateSeatView seatInfo:
    TUISeatInfo, seatView: UIView)
```

参数:

| 参数           | 类型           | 描述        |
|--------------|--------------|-----------|
| seatGridView | SeatGridView | 语聊房核心组件   |
| seatInfo     | TUISeatInfo  | 麦位信息      |
| seatView     | UIView       | 当前更新的麦位视图 |

## seatGridView:updateUserVolume

更新用户音量时回调，您可以根据回调回来的音量更新自己的麦位视图。

```
func seatGridView(_ view: SeatGridView, updateUserVolume volume: CGFloat,
    seatView: UIView)
```

参数:

| 参数           | 类型           | 描述            |
|--------------|--------------|---------------|
| seatGridView | SeatGridView | 语聊房核心组件       |
| volume       | CGFloat      | 麦位音量信息        |
| seatView     | UIView       | 当前音量变化的麦位布局视图 |

# 服务端 API

## 账号系统

## 账号管理

最近更新时间：2025-06-27 17:58:42

TUICallKit、TUIRoomKit、TUILiveKit 的系统 and 即时通信 IM 系统共用一套账号体系，账号即用户名 (UserId)，分为普通账号和管理员账号，登录 [即时通信 IM 控制台](#)，在左侧导航栏选择消息服务 Chat > 账号管理，您可以根据实际业务需求管理账号。

**说明：**  
TUICallKit、TUIRoomKit、TUILiveKit 的系统 and 即时通信 IM 系统共用一套账号体系。

## 新建账号

- 在账号管理页面，单击**新建账号**。
- 在弹出的新建账号对话框中，配置以下参数：

新建账号

账号类型

☒ 普通账号 ☐ 管理员账号 ?

用户名 \*

请输入 UserID

昵称

请输入昵称(非必填)

头像

请输入头像图片链接 URL (非必填)

确定

取消

- 账号类型：区分普通账号和管理员账号。“App管理员”是对 App 具有最高管理权限的角色，可调用 REST API 接口，进行用户管理，房间管理，房间成员管理等操作。每个应用最多支持配置 10 个管理员。
  - 用户名：需输入用户名（UserID），必填。
  - 用户昵称：可输入用户昵称，非必填。
  - 头像：可输入用户头像链接 URL，非必填。
- 单击**确定**保存配置。
  - 账号新建完成后，在账号列表中可以查看用户名、昵称、账号类型、头像和创建时间。



# 删除账号

1. 在账号管理页面，在列表中选中需要删除的账号，单击**批量删除**。



2. 在弹出的确认对话框中，单击**确定**确认删除。删除后数据无法恢复，请谨慎操作。



# 编辑账号

1. 在账号管理页面，在列表中希望编辑的账号操作列，单击**编辑**。

2. 在弹出的编辑账号对话框中，配置以下参数：

编辑账号信息

×

用户名 \*

administrator

昵称

请输入昵称（非必填）

账号类型

管理员账号

"App管理员"是对 App 具有最高管理权限的角色，可调用[REST API接口](#)，进行创建/解散群组、发送全员 / 标签推送消息等操作。每个应用最多支持配置 15 个管理员。

性别

请选择

生日

请输入用户生日，推荐用法 20190419（非必填）

所在地

请输入用户所在地（非必填）

个性签名

请输入用户个性签名（非必填）

加好友验证方式

允许任何人添加自己为好友

语言

请输入用户语言（非必填）

头像

请输入头像图片链接 URL (非必填)

管理员禁止加好友标识

允许加好友

等级

请输入用户等级 (非必填)

角色

请输入用户角色 (非必填)

确定

取消

- 用户名：不可编辑。
- 昵称：可编辑用户昵称，非必填。
- 账号类型：不可编辑。
- 性别：可编辑性别，非必填。
- 生日：可编辑生日，非必填。
- 所在地：可编辑所在地，非必填。
- 个性签名：可编辑个性签名，非必填。
- 语言：可编辑语言，非必填。
- 头像：可编辑头像，非必填。
- 等级：可编辑等级，非必填。
- 角色：可编辑角色，非必填。

2.1 单击**确定**保存配置。

导出账号

1. 单独导出：在账号管理页面，在列表中希望导出的账号操作列，单击**导出**。

| <input type="checkbox"/> | 用户名 (UserID) | 昵称 (NickName) | 账号类型 ▾ | 头像 (Image) | 创建时间                | 操作                                                                             |
|--------------------------|--------------|---------------|--------|------------|---------------------|--------------------------------------------------------------------------------|
| <input type="checkbox"/> | 3184         |               | 普通账号   |            | 2024-09-23 14:34:32 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |
| <input type="checkbox"/> | 522582       |               | 普通账号   |            | 2024-08-21 16:01:47 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |
| <input type="checkbox"/> | 4360         |               | 普通账号   |            | 2024-08-23 10:11:27 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |

2. 批量导出：在账号管理页面，在列表中选中需要导出的账号，单击**批量导出**。

新建账号

批量导入

批量导出

当前版本默认不允许删除账号，[点此解除限制](#)

| <input type="checkbox"/>            | 用户名 (UserID) | 昵称 (NickName) | 账号类型 ▾ | 头像 (Image) | 创建时间                | 操作                                                                             |
|-------------------------------------|--------------|---------------|--------|------------|---------------------|--------------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> | 3184         |               | 普通账号   |            | 2024-09-23 14:34:32 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |
| <input checked="" type="checkbox"/> | 522582       |               | 普通账号   |            | 2024-08-21 16:01:47 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |
| <input type="checkbox"/>            | 4360         |               | 普通账号   |            | 2024-08-23 10:11:27 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |
| <input type="checkbox"/>            | 571408       |               | 普通账号   |            | 2024-08-23 10:15:34 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |

3. 在弹出的导出对话框中提示导出成功后，单击**下载**，即完成导出。



设置/取消管理员账号

1. 设置/取消管理员：在账号管理页面，在列表中找到希望设置或取消管理员的普通账号，单击**设置/取消管理员**。

| <input type="checkbox"/> | 用户名 (UserID) | 昵称 (NickName) | 账号类型 ▾ | 头像 (Image) | 创建时间                | 操作                                                                             |
|--------------------------|--------------|---------------|--------|------------|---------------------|--------------------------------------------------------------------------------|
| <input type="checkbox"/> | 3184         |               | 普通账号   |            | 2024-09-18 17:50:32 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |
| <input type="checkbox"/> | 522582       |               | 普通账号   |            | 2024-08-21 16:01:47 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |
| <input type="checkbox"/> | 4360         |               | 普通账号   |            | 2024-08-23 10:11:27 | <a href="#">导出</a> <a href="#">编辑</a> <a href="#">设置管理员</a> <a href="#">删除</a> |

 **注意：**

administrator 账号不支持取消其管理员身份。

2. 在弹出的确认对话框中，单击**确定**确认即可。



# 登录鉴权

最近更新时间：2025-06-30 12:41:24

## 登录鉴权简介

TUICallKit、TUIRoomKit、TUILiveKit 与即时通信 IM 共用同一套登录鉴权体系，即时通信 IM 的前身是 QQ 的即时通信系统，我们抽离 QQ 的通用模块，并将其整合成适合终端接入的 IM SDK 及后台服务。您可以把 IM SDK 理解为一个没有用户交互界面的 QQ，把 IM SDK 集成到您的 App 里，就相当于把一个 QQ 的内核集成在您的 App 内部。

QQ 可以用来收发消息，但前提是您必须先登录才能使用。登录 QQ 用的是 QQ 号和密码，登录 IM SDK 需要使用您指定的用户名（UserID）和密码（UserSig）。

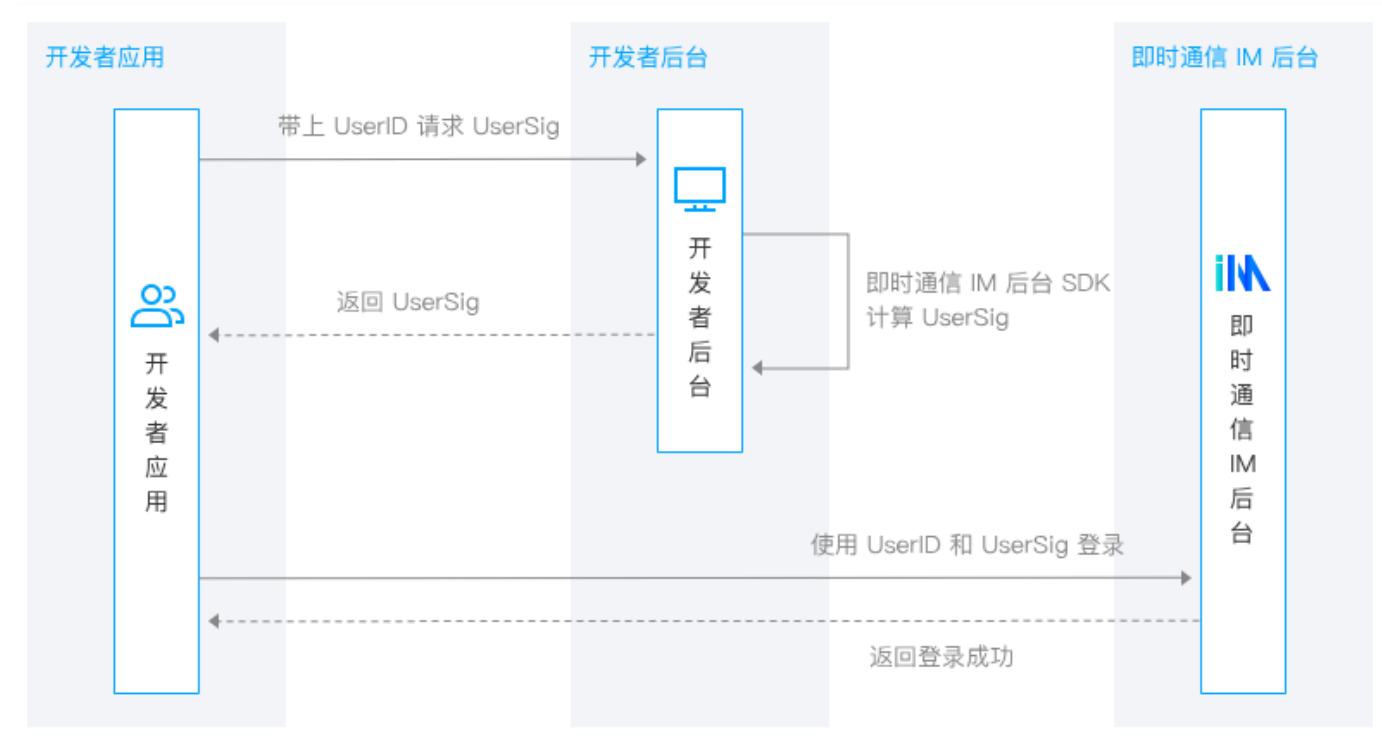
- **UserID**：旧称为 Identifier，用户登录即时通信 IM 时使用的用户名，即您 App 里的用户 ID。
- 例如，App 里有一个用户，该用户的 ID 是 27149，那么您可以用 27149 作为登录即时通信 IM 的 UserID。
- **UserSig**：用户登录即时通信 IM 时使用的密码，其本质是 App Server 用密钥对 UserID 等信息加密后的数据。具体生成方法请参见 [生成 UserSig](#)。

### 说明：

TUICallKit、TUIRoomKit、TUILiveKit 的系统 and 即时通信 IM 系统共用一套登录鉴权体系。

## App 登录流程

推荐的 App 登录即时通信 IM 流程如下：



**⚠ 注意：**

- 即时通信 IM 后台完全信赖 UserSig，为避免数据和业务受影响，请务必确保私钥的安全。
- 即时通信 IM 后台 SDK 默认接口生成的 UserSig 有效期为180天，可以使用含有效期参数的接口自行设定有效期，开发者需要在 UserSig 过期前到开发者后台获取新的 UserSig。
- 用于生成 UserSig 的即时通信 IM 后台 SDK 详细请参阅 [生成 UserSig](#)。

## App 管理员

即时通信 IM 的部分服务需要管理员权限，例如调用 REST API 接口、解散群以及全员推送等。App 管理员是对 App 具有最高管理权限的角色，与普通账号相比，其区别如下：

- 读取权限更高。例如，获取 App 内部的所有群组、获取任意群组的任意资料。
- 操作权限更高。例如，给任意用户发消息、在任意群组中增删成员。

设置 App 管理员仅能通过登录控制台操作，具体操作请参见 [配置账号管理员](#)。

# 生成 UserSig

最近更新时间：2025-06-27 17:58:42

UserSig 是用户登录的密码，也是使用RestAPI的签名，其本质是对 UserID 等信息加密后得到的密文，本文将指导您如何生成 UserSig。以下视频将帮助您快速了解如何生成 UserSig：

**说明：**  
TUICallKit、TUIRoomKit、TUILiveKit 的系统 and 即时通信 IM 系统共用同一套 Usersig 的生成方式。

[观看视频](#)

## 获取密钥

1. 登录即时通信 IM [控制台](#)。

**说明：**  
如果您还没有应用，请先通过 [开通服务](#) 创建一个应用。

2. 单击[应用管理](#)，进入应用管理列表。

3. 在目标应用的操作列，单击[查看密钥](#)。



| 应用名称 | SDKAppID | 应用状态 | 数据中心 | 标签 | 创建时间       | 服务信息                                                       | 服务状态 | 到期时间 | 操作                                        |
|------|----------|------|------|----|------------|------------------------------------------------------------|------|------|-------------------------------------------|
| ...  | 16...    | 正常   | 中国   | -  | 2024-03-21 | <a href="#">消息服务 Chat 体验版</a><br><a href="#">推送服务 Push</a> | 使用中  | -    | <a href="#">应用详情</a> <a href="#">查看密钥</a> |
| ...  | 16...    | 正常   | 中国   | -  | 2023-10-16 | <a href="#">消息服务 Chat 体验版</a><br><a href="#">推送服务 Push</a> | 使用中  | -    | <a href="#">应用详情</a> <a href="#">查看密钥</a> |

4. 在对话框中单击[显示密钥](#)。



### 查看密钥

密钥信息为敏感信息，请注意保密，不要泄露。

密钥 \*\*\*\*\*

[显示密钥](#)

5. 单击复制图标即可复制并储存密钥信息。

**⚠ 注意：**

请妥善保管密钥信息，谨防泄露。

## 客户端计算 UserSig

IM SDK 示例代码中提供的 `GenerateTestUserSig` 的开源模块可以帮忙您快速生成 UserSig。您只需设置 SDKAPPID（应用 SDKAppID）、EXPIRETIME（UserSig 过期时间）和 SECRETKEY（密钥信息）三个成员变量的取值，然后调用 `genTestUserSig()` 函数即可快速获取 UserSig。

为了简化实现过程，我们提供了下列语言或平台的 UserSig 计算源码，您可以直接下载并集成到您的客户端。

| 编程语言        | 所属平台    | GenerateTestUserSig 源代码                  |
|-------------|---------|------------------------------------------|
| Java        | Android | <a href="#">GenerateTestUserSig.java</a> |
| Objective-C | iOS     | <a href="#">GenerateTestUserSig.h</a>    |
| Objective-C | Mac     | <a href="#">GenerateTestUserSig.h</a>    |
| C++         | Windows | <a href="#">GenerateTestUserSig.h</a>    |
| Javascript  | Web     | <a href="#">GenerateTestUserSig.js</a>   |
| Javascript  | 小程序     | <a href="#">GenerateTestUserSig.js</a>   |
| Dart        | Flutter | <a href="#">GenerateTestUserSig.dart</a> |

**⚠ 注意：**

该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此该方法仅适合本地跑通 Demo 和功能调试。

正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 [服务端生成 UserSig](#)。



## 服务端计算 UserSig

采用服务端计算 UserSig，可以最大限度地保障计算 UserSig 所用的密钥信息不被泄露。您只需将计算代码部署在您的服务器上，并提供面向 App 的服务端接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。

为了简化实现过程，我们提供了下列语言或平台的 UserSig 计算源码，您可以直接下载并集成到您的服务端。

| 语言版本   | 关键函数        | 下载链接                    |
|--------|-------------|-------------------------|
| Java   | HMAC-SHA256 | <a href="#">genSig</a>  |
| GO     | HMAC-SHA256 | <a href="#">GenSig</a>  |
| PHP    | HMAC-SHA256 | <a href="#">genSig</a>  |
| Nodejs | HMAC-SHA256 | <a href="#">genSig</a>  |
| Python | HMAC-SHA256 | <a href="#">gen_sig</a> |
| C#     | HMAC-SHA256 | <a href="#">GenSig</a>  |
| C++    | HMAC-SHA256 | <a href="#">gen_sig</a> |

UserSig 计算函数中主要包括 SDKAppID、UserID 以及 UserSig 有效期等关键参数，关键参数的详细介绍如下表所示：

❗ 说明：

下表中的字段名以 Java 语言的源码为例，其他语言略有差异，请以实际字段名为准。

| 字段名示例    | 参数说明                                                                               |
|----------|------------------------------------------------------------------------------------|
| sdkappid | 应用 SDKAppID，可在即时通信 IM <a href="#">控制台</a> 的应用卡片中获取。                                |
| userId   | 用户 ID，旧称为 Identifier。                                                              |
| expire   | UserSig 的有效期，单位为秒。建议 UserSig 有效期最短不小于24小时，最长不超过50年。为了您的账号安全，建议将 UserSig 有效期设置为两个月。 |
| userbuf  | 即时通信 IM 中均默认使用不带 UserBuf 的接口，即该参数默认填写为 <code>null</code> 。                         |
| key      | 密钥信息，可在即时通信 IM <a href="#">控制台</a> 的应用详情页面中获取。                                     |

## 老版本算法

为了简化签名计算难度，方便客户更快速地使用腾讯云服务，即时通信 IM 服务自2019.07.19开始启用新的签名算法，从之前的 ECDSA-SHA256 升级为 HMAC-SHA256。2019.07.19以后创建的 SDKAppID 均会采用新的 HMAC-SHA256 算法。

如果您的 SDKAppID 是2019.07.19之前创建的，建议升级为 HMAC-SHA256 算法，升级过程不会影响现网业务。您也可以继续使用老版本的签名算法，ECDSA-SHA256 算法的源码下载链接如下：

| 语言版本   | 签名算法         | 下载链接                   |
|--------|--------------|------------------------|
| Java   | ECDSA-SHA256 | <a href="#">Github</a> |
| GO     | ECDSA-SHA256 | <a href="#">Github</a> |
| PHP    | ECDSA-SHA256 | <a href="#">Github</a> |
| Nodejs | ECDSA-SHA256 | <a href="#">Github</a> |
| Python | ECDSA-SHA256 | <a href="#">Github</a> |
| C#     | ECDSA-SHA256 | <a href="#">Github</a> |
| C++    | ECDSA-SHA256 | <a href="#">Github</a> |

# 资料管理

最近更新时间：2025-06-27 17:58:42

TUICallKit、TUIRoomKit、TUILiveKit 的系统和即时通信 IM 系统共用同一套账号体系，资料属于账号体系里用户绑定的一些信息。

## 资料系统简介

即时通信 IM 开放了用户资料托管能力，提供资料相关的一整套完整解决方案。如果您想让每个用户都有自己的资料，并且轻松实现资料的设置和拉取功能，那么您可以选择即时通信 IM 的资料托管服务：

- 即时通信 IM 提供资料的存储能力，确保您的数据具备异地容灾、多地部署和自动扩容/缩容的能力，帮助您从服务器宕机、多拷贝主从复制和扩容缩容等复杂处理流程中得到完全地解放。
- 即时通信 IM 提供业界通用的业务处理流程，帮助您在用户资料的业务逻辑上彻底地解放。
- 即时通信 IM 提供专业的运营流程和运营团队，全年99.99%的稳定服务质量，帮助您为用户提供具有稳定口碑的服务。
- 即时通信 IM 提供简单易用的服务接口和快捷接入的帮助指引，全程为您提供星级服务。

使用即时通信 IM 的资料托管服务，可以得到以下能力：

- 标配资料字段的存储、读写能力。
- 自定义资料字段的存储、读写能力。

## 资料字段

资料是用来描述用户属性的一组数据，即时通信 IM 资料系统支持标配资料字段和自定义资料字段。资料字段有如下特性：

- 资料字段通过 Key-Value 来表示。
- Key 为 String 类型，其命名仅支持英文大小写字母、数字、下划线。
- Value 有以下几种类型：
  - a. uint32\_t 类型的整数（自定义资料字段不支持）。
  - b. uint64\_t 类型的整数（自定义资料字段不支持）。
  - c. string 类型的字符串（string 的长度不得超过500字节）。
  - d. bytes 类型的一段 buffer（buffer 的长度不得超过500字节）。

## 标配资料字段

目前即时通信 IM 支持的标配资料字段如下：

| 字段名称          | 类型     | 描述 | 更新时有 Push | 备注           |
|---------------|--------|----|-----------|--------------|
| Tag_Profile_I | String | 昵称 | 有         | 长度不得超过500个字节 |

|                          |        |           |   |                                                 |
|--------------------------|--------|-----------|---|-------------------------------------------------|
| M_Nick                   |        |           |   |                                                 |
| Tag_Profile_I<br>M_Image | String | 头像<br>URL | 有 | 长度不得超过500个字节                                    |
| Tag_Profile_I<br>M_Level | uint32 | 等级        | 有 | 通常一个 UINT-8 数据即可保存一个等级信息您可以考虑拆分保存，从而实现多种角色的等级信息 |
| Tag_Profile_I<br>M_Role  | uint32 | 角色        | 有 | 通常一个 UINT-8 数据即可保存一个角色信息，您可以考虑拆分保存，从而保存多种角色信息   |

## 自定义资料字段

自定义资料字段是各 App 根据各自业务需要而设置的用户数据。通过自定义资料，各 App 可以将一些额外数据附加到用户资料上，并通过现有接口进行读写操作。

### 自定义资料字段的申请

App 管理员可以通过即时通信 IM [控制台](#) > [应用配置](#) > [功能配置](#) 申请自定义资料字段，申请提交后，自定义资料字段将在10分钟内生效。

申请自定义资料字段时，需要为每一个自定义资料字段提交如下资料：

- 自定义资料字段的名称（Key）：详情可参见 [自定义资料字段的命名规范](#)。
- 自定义资料字段的类型（Value）：详情可参见 [资料字段](#)。

### 自定义资料字段的命名规范

自定义资料字段的命名规则如下：

- 自定义资料字段的名称包括前缀和关键字两部分。
- 自定义资料字段的前缀是：Tag\_Profile\_Custom。
- 关键字：必须是英文字母，且长度不得超过8字节，建议用一个英文单词或该英文单词的缩写。
- 示例：某 App 要申请的自定义字段的关键字是 Test，则自定义资料字段的名称是：Tag\_Profile\_Custom\_Test。

# REST API

## REST API 简介

最近更新时间：2025-04-03 17:08:43

REST API 是在线直播房间 SDK 的后台 HTTP 管理接口，其主要目的在于为开发者提供一套简单的管理入口。为了安全性，REST API 仅提供 HTTPS 接口。

### 前提条件

要调用 REST API，您必须开通 TUILiveKit 服务。

### 调用方法

#### 请求 URL

REST API 的 URL 格式如下：

```
https://xxxxxx/$ver/$servicename/$command?
sdkappid=$SDKAppID&identifier=$identifier&usersig=$usersig&random=99999999
&contenttype=json
```

其中各个参数的含义以及取值如下（参数名称及其取值均区分大小写）：

| 参数          | 含义                              | 取值                                                                                                                                                  |
|-------------|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| https       | 请求协议                            | 请求协议为 HTTPS，请求方式为 POST                                                                                                                              |
| xxxxxx      | 请求域名                            | <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| ver         | 协议版本号                           | 固定为 v4                                                                                                                                              |
| servicename | 内部服务名，不同的 servicename 对应不同的服务类型 | 示例：<br>v4/live_engine_http_srv/create_room<br>，其中 room_engine_http_srv 为 servicename 更多详情请参见 <a href="#">REST API 列表</a>                            |

|             |                                 |                                                                                                             |
|-------------|---------------------------------|-------------------------------------------------------------------------------------------------------------|
| command     | 命令字，与 servicename 组合用来标识具体的业务功能 | 示例：<br>v4/live_engine_http_srv/create_room<br>，其中 create_room 为 command 更多详情请参见 <a href="#">REST API 列表</a> |
| sdkappid    | App 在即时通信 IM 控制台获取的应用标识         | 在申请接入时获得                                                                                                    |
| identifier  | 用户名，调用 REST API 时必须为 App 管理员账号  | 参见 <a href="#">App 管理员</a>                                                                                  |
| usersig     | 用户名对应的密码                        | 参见 <a href="#">生成 UserSig</a>                                                                               |
| random      | 标识当前请求的随机数参数                    | 32位无符号整数随机数，取值范围0 - 4294967295                                                                              |
| contenttype | 请求格式                            | 固定值为 json                                                                                                   |

#### ⚠ 注意：

- App 服务端在调用 REST API 时，identifier 必须为 App 管理员账号。
- App 可以在每次调用 REST API 时都生成管理员账号的 UserSig，亦可生成一个固定的 UserSig 重复使用，但请特别注意 UserSig 的有效期。

## HTTP 请求包体格式

REST API 仅支持 POST 方法，其请求包体为 JSON 格式，具体的包体格式参见每个 API 的详细描述。需要特别注意的是，POST 包体不能为空，即使某条协议包体中不需要携带任何信息，也需要携带一个空的 JSON 对象，即 {}。

## HTTP 返回码

除非发生网络错误（例如502错误），否则 REST API 的调用结果均为200，真正的 API 调用错误码与错误信息在 HTTP 应答包体中返回。

## HTTP 应答包体格式

REST API 的应答包体也是 JSON 格式，其格式符合如下特征：

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-70e312f1de024af5a36714b7b71da224-O-Seq-63504"
```

// REST API 其他应答内容

}

应答包体中必然包含 `ActionStatus`、`ErrorInfo`、`ErrorCode`、`RequestId` 这三个属性，其含义如下：

| 字段                        | 类型      | 说明                                                                  |
|---------------------------|---------|---------------------------------------------------------------------|
| <code>ActionStatus</code> | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败，如果为 FAIL， <code>ErrorInfo</code> 带上失败原因 |
| <code>ErrorInfo</code>    | String  | 失败原因                                                                |
| <code>ErrorCode</code>    | Integer | 错误码，0为成功，其他为失败，可查询 <a href="#">错误码表</a> 得到具体的原因                     |
| <code>RequestId</code>    | String  | 错误码，0为成功，其他为失败，可查询 <a href="#">错误码表</a> 得到具体的原因                     |

## REST API 公共错误码

| 错误码   | 含义说明                                      |
|-------|-------------------------------------------|
| 60002 | HTTP 解析错误，请检查 HTTP 请求 URL 格式              |
| 60003 | HTTP 请求 JSON 解析错误，请检查 JSON 格式             |
| 60004 | 请求 URL 或 JSON 包体中账号或签名错误                  |
| 60005 | 请求 URL 或 JSON 包体中账号或签名错误                  |
| 60006 | SDKAppID 失效，请核对 SDKAppID 有效性              |
| 60007 | REST 接口调用频率超过限制，请降低请求频率                   |
| 60008 | 服务请求超时或 HTTP 请求格式错误，请检查并重试                |
| 60009 | 请求资源错误，请检查请求 URL                          |
| 60010 | 请求需要 App 管理员权限                            |
| 60011 | SDKAppID 请求频率超限，请降低请求频率                   |
| 60012 | REST 接口需要带 SDKAppID，请检查请求 URL 中的 SDKAppID |
| 60013 | HTTP 响应包 JSON 解析错误                        |
| 60014 | 置换账号超时                                    |
| 60015 | 请求包体账号类型错误，请确认账号为字符串格式                    |

|       |                                                                                                                                               |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 60016 | SDKAppID 被禁用。                                                                                                                                 |
| 60017 | 请求被禁用。                                                                                                                                        |
| 60018 | 请求过于频繁，请稍后重试。                                                                                                                                 |
| 60019 | 请求过于频繁，请稍后重试。                                                                                                                                 |
| 60020 | 您的专业版套餐包已到期并停用，请登录 [即时通信 IM 购买页面] ( <a href="https://buy.cloud.tencent.com/avc">https://buy.cloud.tencent.com/avc</a> ) 重新购买套餐包。购买后，将在5分钟后生效。 |
| 60021 | RestAPI 调用来源 IP 非法。                                                                                                                           |

## FAQ

### REST API 请求有概率超时，收不到任何响应？

1. Live 后台 REST 接口设置的超时时间是3s，调用方设置的超时时间应该长于3s。
2. `telnet console.tim.qq.com 443` 确认能否连接服务端口。
3. 使用 `curl -I https://console.tim.qq.com` 简单测试看状态码是否为200。
4. 确认机器的 dns server 配置是内网 dns server，还是公共 dns server。如果是内网 dns server，请确保 dns server 网络出口和本机器网络出口 IP 所在地域运营商一致。
5. 建议业务调用方使用“长连接+连接池”模式。

#### ❗ 说明：

由于 HTTPS 短连接建连耗时比较大，每次请求都有 TCP + tls 握手开销，所以建议 REST API 长连接接入。

使用标准 HTTP 库的场景：HTTP1.0 需要指定请求头部 `Connection: keep-alive`，HTTP1.1 默认支持长连接；基于 TCP 封装 HTTPS 请求的场景，可以复用 TCP 连接来收发请求。



# REST API 列表

最近更新时间：2025-04-03 17:08:43

## 房间管理

| 功能说明      | 接口                                                     |
|-----------|--------------------------------------------------------|
| 创建房间      | <a href="#">live_engine_http_srv/create_room</a>       |
| 解散房间      | <a href="#">live_engine_http_srv/destroy_room</a>      |
| 更新房间信息    | <a href="#">live_engine_http_srv/update_room_info</a>  |
| 获取房间信息    | <a href="#">live_engine_http_srv/get_room_info</a>     |
| 获取直播列表    | <a href="#">live_engine_http_srv/get_room_list</a>     |
| 获取房间自定义信息 | <a href="#">live_engine_http_srv/get_room_metadata</a> |
| 设置房间自定义信息 | <a href="#">live_engine_http_srv/set_room_metadata</a> |
| 删除房间自定义信息 | <a href="#">live_engine_http_srv/del_room_metadata</a> |

## 房间成员管理

| 功能说明     | 接口                                                          |
|----------|-------------------------------------------------------------|
| 获取管理员列表  | <a href="#">live_engine_http_srv/get_admin_list</a>         |
| 成员封禁     | <a href="#">live_engine_http_srv/ban_member</a>             |
| 成员解封     | <a href="#">live_engine_http_srv/unban_member</a>           |
| 获取封禁列表   | <a href="#">live_engine_http_srv/get_banned_member_list</a> |
| 修改管理员    | <a href="#">live_engine_http_srv/modify_admin</a>           |
| 获取管理员列表  | <a href="#">live_engine_http_srv/get_admin_list</a>         |
| 成员禁言     | <a href="#">live_engine_http_srv/mute_member</a>            |
| 获取禁言列表   | <a href="#">live_engine_http_srv/get_muted_member_list</a>  |
| 获取房间成员列表 | <a href="#">live_engine_http_srv/get_member_list</a>        |

## 账号管理

| 功能说明   | 接口                                                    |
|--------|-------------------------------------------------------|
| 导入单个账号 | <a href="#">im_open_login_svc/account_import</a>      |
| 导入多个账号 | <a href="#">im_open_login_svc/multiaccount_import</a> |
| 查询账号   | <a href="#">im_open_login_svc/account_check</a>       |
| 设置用户资料 | <a href="#">profile/portrait_set</a>                  |
| 拉取用户资料 | <a href="#">profile/portrait_get</a>                  |

# 用户管理

## 账号管理

### 导入单个账号

最近更新时间：2025-04-03 17:08:43

#### 功能说明

本接口用于将 App 自有账号导入 Live 账号系统，为该账号创建一个对应的内部 ID，使该账号能够使用 Live 服务。

 **注意：**  
同一个账号重复导入仅会创建1个内部 ID，Live 和即时通信 IM 共用一套底层账号系统。

以下视频将帮助您快速了解如何将 App 自有账号导入 Live 账号系统：

[观看视频](#)

#### 接口调用说明

#### 请求 URL 示例

```
https://xxxxxx/v4/im_open_login_svc/account_import?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

#### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 [REST API 简介](#)。

| 参数                        | 说明                                                                                                                                                                                                                                                                                                          |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                    | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国： console.tim.qq.com</li><li>● 新加坡： adminapisgp.im.qcloud.com</li><li>● 首尔： adminapikr.im.qcloud.com</li><li>● 法兰克福： adminapiger.im.qcloud.com</li><li>● 美国： adminapiusa.im.qcloud.com</li><li>● 印尼： adminapiidn.im.qcloud.com</li></ul> |
| v4/im_open_login_svc/acco | 请求接口                                                                                                                                                                                                                                                                                                        |

|             |                                                   |
|-------------|---------------------------------------------------|
| unt_import  |                                                   |
| sdkappid    | 创建应用时即时通信 IM 控制台分配的 SDKAppID                      |
| identifier  | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>     |
| usersig     | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a> |
| random      | 请输入随机的32位无符号整数，取值范围0 – 4294967295                 |
| contenttype | 请求格式固定值为 <code>json</code>                        |

## 最高调用频率

200次/秒。

## 请求包示例

```
{
  "UserID": "test",
  "Nick": "Nick_001",
  "FaceUrl": "http://www.qq.com"
}
```

## 请求包字段说明

| 字段      | 类型     | 属性 | 说明                                                    |
|---------|--------|----|-------------------------------------------------------|
| UserID  | String | 必填 | 用户名，长度不超过32字节                                         |
| Nick    | String | 选填 | 用户昵称，对应于 <a href="#">资料字段</a> 里的 Tag_Profile_IM_Nick  |
| FaceUrl | String | 选填 | 用户头像，对应于 <a href="#">资料字段</a> 里的 Tag_Profile_IM_Image |

## 应答包体示例

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                                                                                   |
|--------------|---------|--------------------------------------------------------------------------------------|
| ActionStatus | String  | 请求处理的结果： <ul style="list-style-type: none"><li>OK 表示处理成功</li><li>FAIL 表示失败</li></ul> |
| ErrorInfo    | String  | 错误信息                                                                                 |
| ErrorCode    | Integer | 错误码： <ul style="list-style-type: none"><li>0表示成功</li><li>非0表示失败</li></ul>            |

### 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。  
公共错误码（60000到79999）参见 [错误码](#) 文档。  
本 API 私有错误码如下：

| 错误码   | 含义说明                             |
|-------|----------------------------------|
| 40006 | 设置资料时服务器内部错误，请稍后重试               |
| 40601 | 资料字段的 Value 长度超过500字节            |
| 70169 | 服务端内部超时，请稍后重试                    |
| 70398 | 账号数超限。                           |
| 70399 | 非 IM 体验版应用，账号被删除后，三个月内不允许重新导入    |
| 70402 | 参数非法，请检查必填字段是否填充，或者字段的填充是否满足协议要求 |
| 70403 | 请求失败，需要 App 管理员权限                |
| 70500 | 服务器内部错误，请稍后重试                    |

### 接口调试工具

通过 [REST API 在线调试](#) 工具调试本接口。

# 导入多个账号

最近更新时间：2025-04-03 17:08:43

## 功能说明

本接口用于批量将 App 自有账号导入 Live 账号系统，为该账号创建一个对应的内部 ID，使该账号能够使用 Live 服务。

-  **说明：**
- 本接口单次最多支持导入100个账号，同一个账号重复导入仅会创建1个内部 ID。
  - 批量导入账号时，如果账号已存在，会按新传入的参数（用户昵称、头像 URL 等）进行更新。
  - Live 系统和 即时通信 IM系统共用一套账号体系。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/im_open_login_svc/multiaccount_import?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 [REST API 简介](#)。

| 参数                                       | 说明                                                                                                                                                                                                                                                                                                          |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxxx                                  | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国： console.tim.qq.com</li><li>● 新加坡： adminapisgp.im.qcloud.com</li><li>● 首尔： adminapikr.im.qcloud.com</li><li>● 法兰克福： adminapiger.im.qcloud.com</li><li>● 美国： adminapiusa.im.qcloud.com</li><li>● 印尼： adminapiidn.im.qcloud.com</li></ul> |
| v4/im_open_login_svc/multiaccount_import | 请求接口                                                                                                                                                                                                                                                                                                        |
| sdkappid                                 | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                                                                                                                                                |

|             |                                                   |
|-------------|---------------------------------------------------|
| identifier  | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>     |
| usersig     | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a> |
| random      | 请输入随机的32位无符号整数，取值范围0 – 4294967295                 |
| contenttype | 请求格式固定值为 <code>json</code>                        |

## 最高调用频率

100次/秒。

## 新版本请求包示例

```
{
  "AccountList": [
    {
      "UserID": "UserID_001",
      "Nick": "Nick_001",
      "FaceUrl": "http://www.qq.com"
    },
    {
      "UserID": "UserID_002",
      "Nick": "Nick_002",
      "FaceUrl": "http://www.qq.com"
    },
    {
      "UserID": "UserID_002",
      "Nick": "Nick_001",
      "FaceUrl": "http://www.qq.com"
    }
  ]
}
```

## 新版本请求包字段说明

| 字段          | 类型     | 属性 | 说明                                                   |
|-------------|--------|----|------------------------------------------------------|
| AccountList | Array  | 必填 | 待导入的用户列表，单次最多导入100个用户                                |
| UserID      | String | 必填 | 用户的 UserID，UserID 长度不超过32字节                          |
| Nick        | String | 选填 | 用户昵称，对应于 <a href="#">资料字段</a> 里的 Tag_Profile_IM_Nick |

|         |        |    |                                                       |
|---------|--------|----|-------------------------------------------------------|
| FaceUrl | String | 选填 | 用户头像，对应于 <a href="#">资料字段</a> 里的 Tag_Profile_IM_Image |
|---------|--------|----|-------------------------------------------------------|

## 应答包体示例

```
{
  "ActionStatus": "OK",
  "ErrorCode": 0,
  "ErrorInfo": "",
  "FailAccounts": [
    "UserID_002",
    "UserID_003"
  ]
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                                                                                     |
|--------------|---------|----------------------------------------------------------------------------------------|
| ActionStatus | String  | 请求处理的结果 <ul style="list-style-type: none"> <li>OK 表示处理成功</li> <li>FAIL 表示失败</li> </ul> |
| ErrorCode    | Integer | 错误码： <ul style="list-style-type: none"> <li>0表示成功</li> <li>非0表示失败</li> </ul>           |
| ErrorInfo    | String  | 错误信息                                                                                   |
| FailAccounts | Array   | 导入失败的账号列表                                                                              |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码   | 含义说明               |
|-------|--------------------|
| 40006 | 设置资料时服务器内部错误，请稍后重试 |



|       |                                  |
|-------|----------------------------------|
| 40601 | 资料字段的 Value 长度超过500字节            |
| 70169 | 服务端内部超时，请稍后重试                    |
| 70202 | 服务端内部超时，请稍后重试                    |
| 70398 | 账号数超限                            |
| 70399 | 非 IM 体验版应用，账号被删除后，三个月内不允许重新导入    |
| 70402 | 参数非法，请检查必填字段是否填充，或者字段的填充是否满足协议要求 |
| 70403 | 请求失败，需要 App 管理员权限                |
| 70500 | 服务器内部错误，请稍后重试                    |

## 接口调试工具

通过 [REST API 在线调试](#) 工具调试本接口。

# 查询账号

最近更新时间：2025-04-03 17:08:43

## 功能说明

用于查询自有账号是否已导入 Live 系统，支持批量查询。

!

说明：

Live 系统和即时通信 IM 系统共用一套账号体系。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/im_open_login_svc/account_check?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 [REST API 简介](#)。

| 参数                                 | 说明                                                                                                                                                                                                                                                                                                         |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                             | <div>SDKAppID 所在国家/地区对应的专属域名：</div> <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>首尔： adminapikr.im.qcloud.com</li><li>法兰克福： adminapiger.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li><li>印尼： adminapiidn.im.qcloud.com</li></ul> |
| v4/im_open_login_svc/account_check | 请求接口                                                                                                                                                                                                                                                                                                       |
| sdkappid                           | 创建应用时控制台分配的 SDKAppID                                                                                                                                                                                                                                                                                       |
| identifier                         | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                                                                                                                                              |
| usersig                            | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                                                                                                                                                          |

|             |                                   |
|-------------|-----------------------------------|
| random      | 请输入随机的32位无符号整数，取值范围0 – 4294967295 |
| contenttype | 请求格式固定值为 json                     |

## 最高调用频率

100次/秒。

## 请求包示例

```
{
  "CheckItem":
  [
    {
      "UserID": "UserID_1"
    },
    {
      "UserID": "UserID_2"
    }
  ]
}
```

## 请求包字段说明

| 字段        | 类型     | 属性 | 说明                         |
|-----------|--------|----|----------------------------|
| CheckItem | Array  | 必填 | 请求检查的账号对象数组，单次请求最多支持100个账号 |
| UserID    | String | 必填 | 请求检查的账号的 UserID            |

## 应答包体示例

```
{
  "ActionStatus": "OK",
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ResultItem": [
    {
      "UserID": "UserID_1",
      "ResultCode": 0,
      "ResultInfo": "",
      "AccountStatus": "Imported"
    },
  ],
}
```

```
{
  "UserID": "UserID_2",
  "ResultCode": 0,
  "ResultInfo": "",
  "AccountStatus": "NotImported"
}
```

应答包字段说明

| 字段            | 类型      | 说明                                                                                   |
|---------------|---------|--------------------------------------------------------------------------------------|
| ActionStatus  | String  | 请求的处理结果： <ul style="list-style-type: none"><li>OK：表示处理成功</li><li>FAIL：表示失败</li></ul> |
| ErrorCode     | Integer | 错误码： <ul style="list-style-type: none"><li>0：表示成功</li><li>非0：表示失败</li></ul>          |
| ErrorInfo     | String  | 请求处理失败时的错误信息                                                                         |
| ResultItem    | Array   | 单个账号的检查结果对象数组                                                                        |
| UserID        | String  | 请求检查的账号的 UserID                                                                      |
| ResultCode    | Integer | 单个账号的检查结果： <ul style="list-style-type: none"><li>0：表示成功</li><li>非0：表示失败</li></ul>    |
| ResultInfo    | String  | 单个账号检查失败时的错误描述信息                                                                     |
| AccountStatus | String  | 单个账号的导入状态：Imported 表示已导入，NotImported 表示未导入                                           |

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。  
公共错误码（60000到79999）请参见 [错误码](#)。  
本 API 私有错误码如下：

| 错误码 | 含义说明 |
|-----|------|
|-----|------|

|       |                                  |
|-------|----------------------------------|
| 70169 | 服务端内部超时，请稍后重试                    |
| 70202 | 服务端内部超时，请稍后重试                    |
| 70402 | 参数非法，请检查必填字段是否填充，或者字段的填充是否符合协议要求 |
| 70403 | 请求失败，需要 App 管理员权限                |
| 70500 | 服务器内部错误，请稍后重试                    |

## 接口调试工具

通过 [REST API 在线调试](#) 工具调试本接口。

# 用户资料 设置资料

最近更新时间：2025-04-03 17:08:43

## 功能说明

资料是绑定到账号上的信息, 支持 [标配资料字段](#) 和 [自定义资料字段](#) 的设置。

 **说明：**  
Live 和 即时通信 IM 共用同一套账号体系，因此账号附带的资料也是共用的。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/profile/portrait_set?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype  
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 [REST API 简介](#)。

| 参数                      | 说明                                                                                                                                                                                                                                                                                                          |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                  | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国： console.tim.qq.com</li><li>● 新加坡： adminapisgp.im.qcloud.com</li><li>● 首尔： adminapikr.im.qcloud.com</li><li>● 法兰克福： adminapiger.im.qcloud.com</li><li>● 美国： adminapiusa.im.qcloud.com</li><li>● 印尼： adminapiidn.im.qcloud.com</li></ul> |
| v4/profile/portrait_set | 请求接口                                                                                                                                                                                                                                                                                                        |
| sdkappid                | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                                                                                                                                                |
| identifier              | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                                                                                                                                               |
| usersig                 | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                                                                                                                                                           |

|             |                                   |
|-------------|-----------------------------------|
| random      | 请输入随机的32位无符号整数，取值范围0 – 4294967295 |
| contenttype | 请求格式固定值为 <code>json</code>        |

## 请求频率限制

默认接口请求频率限制：200次/秒。

## 请求包示例

```
{
  "From_Account": "id",
  "ProfileItem": [
    {
      "Tag": "Tag_Profile_IM_Nick",
      "Value": "MyNickName"
    }
  ]
}
```

## 请求包字段说明

| 字段           | 类型                    | 属性 | 说明                                     |
|--------------|-----------------------|----|----------------------------------------|
| From_Account | String                | 必填 | 需要设置该 UserID 的资料                       |
| ProfileItem  | Array                 | 必填 | 待设置的用户的资料对象数组，数组中每一个对象都包含了 Tag 和 Value |
| Tag          | String                | 必填 | 指定要设置的资料字段的名称， <a href="#">资料字段</a>    |
| Value        | uint64_t/string/bytes | 必填 | 待设置的资料字段的值                             |

## 应答包体示例

```
{
  "ActionStatus": "FAIL",
  "ErrorCode": 40001,
  "ErrorInfo": "Err_Profile_Comm_Decode_Fail",
  "ErrorDisplay": ""
}
```

```
}

```

### 应答包字段说明

| 字段           | 类型      | 说明                                                                                   |
|--------------|---------|--------------------------------------------------------------------------------------|
| ActionStatus | String  | 请求处理的结果： <ul style="list-style-type: none"><li>OK：表示处理成功</li><li>FAIL：表示失败</li></ul> |
| ErrorCode    | Integer | 错误码： <ul style="list-style-type: none"><li>0：表示成功</li><li>非0：表示失败</li></ul>          |
| ErrorInfo    | String  | 详细错误信息                                                                               |
| ErrorDisplay | String  | 详细的客户端展示信息                                                                           |

### 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码   | 描述                                             |
|-------|------------------------------------------------|
| 40001 | 请求参数错误，请根据错误描述检查请求参数                           |
| 40003 | 请求的用户账号不存在                                     |
| 40004 | 请求需要 App 管理员权限                                 |
| 40006 | 服务器内部错误，请稍后重试                                  |
| 40008 | 没有资料字段的写权限                                     |
| 40009 | 资料字段的 Tag 不存在                                  |
| 40601 | 资料字段的 Value 长度超过500字节                          |
| 40605 | 标配资料字段的 Value错误，详情可参见 <a href="#">标配资料字段</a>   |
| 40610 | 资料字段的 Value 类型不匹配，详情可参见 <a href="#">标配资料字段</a> |

### 接口调试工具



---

通过 [REST API 在线调试](#) 工具调试本接口。

# 拉取资料

最近更新时间：2025-09-09 10:24:42

## 功能说明

- 支持拉取 [标配资料字段](#) 和 [自定义资料字段](#)。
- 建议每次拉取的用户数不超过100，避免因回包数据量太大导致回包失败。
- 请确保请求中的所有账号都已导入即时通信 IM，如果请求中含有未导入即时通信 IM 的账号，即时通信 IM 后台将会提示错误。

**说明：**  
Live 和 即时通信 IM 共用同一套账号体系，因此账号附带的资料也是共用的。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/profile/portrait_get?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 [REST API 简介](#)。

| 参数                      | 说明                                                                                                                                                                                                                                                                                              |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                  | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>首尔： adminapikr.im.qcloud.com</li><li>法兰克福： adminapiger.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li><li>印尼： adminapiidn.im.qcloud.com</li></ul> |
| v4/profile/portrait_get | 请求接口                                                                                                                                                                                                                                                                                            |
| sdkappid                | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                                                                                                                                    |
| identifier              | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                                                                                                                                   |

|             |                                                   |
|-------------|---------------------------------------------------|
| usersig     | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a> |
| random      | 请输入随机的32位无符号整数，取值范围0 – 4294967295                 |
| contenttype | 请求格式固定值为 <code>json</code>                        |

## 请求频率限制

默认接口请求频率限制：200次/秒。

## 请求包示例

### ● 读一个用户的一个资料字段

```
{
  "To_Account": ["id1"],
  "TagList":
  [
    "Tag_Profile_IM_Nick"
  ]
}
```

### ● 读一个用户的多个资料字段

```
{
  "To_Account": ["id1"],
  "TagList":
  [
    "Tag_Profile_IM_Nick",
    "Tag_Profile_IM_AllowType",
    "Tag_Profile_IM_SelfSignature",
    "Tag_Profile_Custom_Test"
  ]
}
```

### ● 读多个用户的一个资料字段

```
{
  "To_Account": ["id1", "id2", "id3"],
  "TagList":
  [
    "Tag_Profile_IM_Nick"
  ]
}
```

```
}

```

### ● 读多个用户的多个资料字段

```
{
  "To_Account":["id1","id2","id3","id4"],
  "TagList":
  [
    "Tag_Profile_IM_Nick",
    "Tag_Profile_IM_AllowType",
    "Tag_Profile_IM_SelfSignature",
    "Tag_Profile_Custom_Test"
  ]
}
```

## 请求包字段说明

| 字段         | 类型    | 属性 | 说明                                                     |
|------------|-------|----|--------------------------------------------------------|
| To_Account | Array | 必填 | 需要拉取这些 UserID 的资料。注意：每次拉取的用户数不得超过100，避免因回包数据量太大以致回包失败。 |
| TagList    | Array | 必填 | 指定要拉取的资料字段的 Tag， <a href="#">资料字段</a>                  |

## 应答包体示例

### ● 读一个用户的一个资料字段

```
{
  "UserProfileItem":
  [
    {
      "To_Account":"id1",
      "ProfileItem":
      [
        {
          "Tag":"Tag_Profile_IM_Nick",
          "Value":"NickNameTest1"
        }
      ],
      "ResultCode":0,
      "ResultInfo":""
    }
  ]
}
```

```
],
"ActionStatus": "OK",
"ErrorCode": 0,
"ErrorInfo": "",
"ErrorDisplay": ""
}
```

## ● 读一个用户的多个资料字段

```
{
  "UserProfileItem":
  [
    {
      "To_Account": "id1",
      "ProfileItem":
      [
        {
          "Tag": "Tag_Profile_IM_Nick",
          "Value": "NickNameTest1"
        },
        {
          "Tag": "Tag_Profile_IM_AllowType",
          "Value": "AllowType_Type_NeedConfirm"
        },
        {
          "Tag": "Tag_Profile_IM_SelfSignature",
          "Value": "I'm Test1"
        },
        {
          "Tag": "Tag_Profile_Custom_Test",
          "Value": "Custom Data1"
        }
      ],
      "ResultCode": 0,
      "ResultInfo": ""
    }
  ],
  "ActionStatus": "OK",
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ErrorDisplay": ""
}
```

- 读多个用户的一个资料字段

```
{
  "UserProfileItem":
  [
    {
      "To_Account": "id1",
      "ProfileItem":
      [
        {
          "Tag": "Tag_Profile_IM_Nick",
          "Value": "NickNameTest1"
        }
      ],
      "ResultCode": 0,
      "ResultInfo": ""
    },
    {
      "To_Account": "id2",
      "ProfileItem":
      [
        {
          "Tag": "Tag_Profile_IM_Nick",
          "Value": "NickNameTest2"
        }
      ],
      "ResultCode": 0,
      "ResultInfo": ""
    },
    {
      "To_Account": "id3",
      "ProfileItem":
      [
        {
          "Tag": "Tag_Profile_IM_Nick",
          "Value": "NickNameTest3"
        }
      ],
      "ResultCode": 0,
      "ResultInfo": ""
    }
  ],
  "ActionStatus": "OK",
  "ErrorCode": 0,
```

```
"ErrorInfo":"","  
"ErrorDisplay":"","  
}
```

## ● 读多个用户的多个资料字段

```
{  
  "UserProfileItem":  
  [  
    {  
      "To_Account":"id1",  
      "ProfileItem":  
      [  
        {  
          "Tag":"Tag_Profile_IM_Nick",  
          "Value":"NickNameTest1"  
        },  
        {  
          "Tag":"Tag_Profile_IM_AllowType",  
          "Value":"AllowType_Type_NeedConfirm"  
        },  
        {  
          "Tag":"Tag_Profile_IM_SelfSignature",  
          "Value":"I'm Test1"  
        },  
        {  
          "Tag":"Tag_Profile_Custom_Test",  
          "Value":"Custom Data1"  
        }  
      ],  
      "ResultCode":0,  
      "ResultInfo":""  
    },  
    {  
      "To_Account":"id2",  
      "ProfileItem":  
      [  
        {  
          "Tag":"Tag_Profile_IM_Nick",  
          "Value":"NickNameTest2"  
        },  
        {  
          "Tag":"Tag_Profile_IM_AllowType",
```

```

        "Value": "AllowType_Type_DenyAny"
    },
    {
        "Tag": "Tag_Profile_IM_SelfSignature",
        "Value": "I'm Test2"
    },
    {
        "Tag": "Tag_Profile_Custom_Test",
        "Value": "Custom Data2"
    }
],
"ResultCode": 0,
"ResultInfo": ""
},
{
    "To_Account": "id3",
    "ProfileItem": [
        {
            "Tag": "Tag_Profile_IM_Nick",
            "Value": "NickNameTest3"
        },
        {
            "Tag": "Tag_Profile_IM_AllowType",
            "Value": "AllowType_Type_AllowAny"
        },
        {
            "Tag": "Tag_Profile_IM_SelfSignature",
            "Value": "I'm Test3"
        },
        {
            "Tag": "Tag_Profile_Custom_Test",
            "Value": "Custom Data3"
        }
    ],
    "ResultCode": 0,
    "ResultInfo": ""
},
{
    "To_Account": "id4",
    "ResultCode": 40006,
    "ResultInfo": "Err_Profile_PortraitGet_Read_Custom_Data_Fail"
}
],

```



```
"Fail_Account":["id4"],
"ActionStatus":"OK",
"ErrorCode":0,
"ErrorInfo":"",
"ErrorDisplay":""
}
```

## 应答包字段说明

| 字段              | 类型                    | 说明                                                                                       |
|-----------------|-----------------------|------------------------------------------------------------------------------------------|
| UserProfileItem | Array                 | 返回的用户资料结构化信息                                                                             |
| To_Account      | String                | 返回的用户的 UserID                                                                            |
| ProfileItem     | Array                 | 返回的用户的资料对象数组，数组中每一个对象都包含了 Tag 和 Value                                                    |
| Tag             | String                | 返回的资料对象的名称： <a href="#">资料字段</a>                                                         |
| Value           | uint64_t/string/bytes | 拉取的资料对象的值                                                                                |
| ResultCode      | Integer               | To_Account 的处理结果： <ul style="list-style-type: none"><li>0：表示成功</li><li>非0：表示失败</li></ul> |
| ResultInfo      | String                | To_Account 的错误描述信息，成功时该字段为空                                                              |
| Fail_Account    | Array                 | 返回处理失败的用户列表，仅当存在失败用户时才返回该字段                                                              |
| ActionStatus    | String                | 请求处理的结果： <ul style="list-style-type: none"><li>OK：表示处理成功</li><li>FAIL：表示失败</li></ul>     |
| ErrorCode       | Integer               | 错误码： <ul style="list-style-type: none"><li>0：表示成功</li><li>非0：表示失败</li></ul>              |
| ErrorInfo       | String                | 详细错误信息                                                                                   |
| ErrorDisplay    | String                | 详细的客户端展示信息                                                                               |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码   | 描述                     |
|-------|------------------------|
| 40001 | 请求参数错误，请根据错误描述检查请求参数   |
| 40002 | 请求参数错误，没有指定 To_Account |
| 40003 | 请求的用户账号不存在             |
| 40004 | 请求需要 App 管理员权限         |
| 40006 | 服务器内部错误，请稍后重试          |
| 40007 | 没有资料字段的读权限             |
| 40009 | 资料字段的 Tag 不存在          |

## 接口调试工具

通过 [REST API 在线调试](#) 工具调试本接口。

# 房间管理

## 创建房间

最近更新时间：2025-09-09 10:24:42

### 功能说明

App 管理员可以通过该接口创建房间。

### 接口调用说明

#### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/create_room?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                  | 说明                                                                                                                                                                              |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                              | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国：console.tim.qq.com</li><li>● 新加坡：adminapisgp.im.qcloud.com</li><li>● 美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/create_room | 请求接口                                                                                                                                                                            |
| sdkappid                            | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                    |
| identifier                          | 必须为 App 管理员账号，更多详情请参见 <a href="#">App管理员</a>                                                                                                                                    |
| usersig                             | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成UserSig</a>                                                                                                                                |
| random                              | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                               |
| contenttype                         | 请求格式固定值为 json                                                                                                                                                                   |

### 最高调用频率

200次/秒。

## 请求包示例

### 基础形式

```
{
  "RoomInfo":{
    "RoomId":"live-room",
    "RoomType":"Live",
    "RoomName":"live name",
    "Owner_Account":"administrator",
    "MaxMemberCount":400,
    "CoverURL":"cover url",
    "IsMessageDisabled":true,
    "Category":[1, 2, 3],
    "IsPublicVisible": true,
    "IsSeatEnabled": true,
    "TakeSeatMode":"ApplyToTake",
    "MaxSeatCount":8,
    "ActivityStatus":1,
    "IsUnlimitedRoomEnabled":true
  }
}
```

## 请求包字段说明

| 字段                | 类型      | 属性 | 说明                                                        |
|-------------------|---------|----|-----------------------------------------------------------|
| RoomId            | String  | 必填 | 房间 ID，最长48个字节                                             |
| RoomName          | String  | 选填 | 房间名称，默认为房间 ID，最长100个字节                                    |
| RoomType          | String  | 必填 | 房间类型：Live（直播房间）                                           |
| Owner_Account     | String  | 选填 | 房主 ID（需是 <a href="#">已导入</a> 的账号），默认为接口调用者的用户 ID          |
| MaxMemberCount    | Integer | 选填 | 最大房间成员数量，缺省时的默认值：付费套餐包上限，例如体验版是20，如果升级套餐包，需按照修改房间资料修改这个字段 |
| CoverURL          | String  | 选填 | 房间封面，最大200字节                                              |
| IsMessageDisabled | Bool    | 选填 | 全体禁音，默认 false                                             |

|                        |         |    |                                                                                 |
|------------------------|---------|----|---------------------------------------------------------------------------------|
| Category               | Array   | 选填 | 房间类型标志, 业务可以用来作为房间类型标识, 例如游戏、音乐等, 最多支持三个                                        |
| IsPublicVisible        | Bool    | 选填 | 房间公开标识                                                                          |
| IsSeatEnabled          | Bool    | 选填 | 是否支持麦位, 默认不支持                                                                   |
| TakeSeatMode           | String  | 选填 | 当使用麦位能力时候, 需要填写麦位模式: FreeToTake (自由上麦), ApplyToTake (申请上麦)                      |
| MaxSeatCount           | Integer | 选填 | 最大麦位数, 当使用麦位能力时候, 默认为套餐包上限                                                      |
| ActivityStatus         | Integer | 选填 | 直播间活跃状态: 用户自定义标记                                                                |
| IsUnlimitedRoomEnabled | Bool    | 选填 | 开启房间混流, 支持高并发场景, 默认为 false。当设置为 true 时候对应于 sdk 侧 <a href="#">TUILiveKit 视频房</a> |

- ⚠ 注意:
- 用户需要在直播结束以后, 手动调用 [解散房间](#) 接口, 结束本场会议。
  - Live [房间自动解散机制](#)。

## 应答包体示例

### 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "RoomId": "live-room"
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                            |
|--------------|---------|-------------------------------|
| ActionStatus | String  | 请求处理的结果, OK 表示处理成功, FAIL 表示失败 |
| ErrorCode    | Integer | 错误码, 0表示成功, 非0表示失败            |

|           |        |                                           |
|-----------|--------|-------------------------------------------|
| ErrorInfo | String | 错误信息                                      |
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| RoomId    | String | 房间 ID                                     |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                                         |
|--------|----------------------------------------------|
| 100001 | 服务器内部错误，请重试                                  |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确                        |
| 100003 | 房间 ID 已存在，请选择其他房间 ID                         |
| 100007 | 无付费信息，需在控制台购买套餐包                             |
| 100010 | 房间 ID 已被使用，并且操作者为房主，可以直接使用                   |
| 100011 | 房间 ID 已被 IM 占用，可以换一个房间 ID 使用，或者先通过 IM 接口解散该群 |
| 100012 | 创建房间超过频率超限，同一房间 ID 1秒内只能创建一次                 |

## 可能触发的回调

- [创建房间之后回调](#)

# 更新房间

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口更新房间信息。

## 接口调用说明

## 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/update_room_info?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                           | 说明                                                                                                                                                                           |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                       | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/upd<br>ate_room_info | 请求接口                                                                                                                                                                         |
| sdkappid                                     | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                                   | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                                      | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                       | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                                  | 请求格式固定值为 json                                                                                                                                                                |

## 最高调用频率

200次/秒。

## 请求包示例

## 基础形式

```
{
  "RoomInfo" : {
    "RoomId": "live-room",
    "RoomName" : "rname-123",
    "MaxMemberCount" : 300,
    "IsMessageDisabled" : false,
    "CoverURL": "https://xxxx.png",
    "Category": [1,2,3],
    "ActivityStatus":1,
    "IsPublicVisible": false,
    "TakeSeatMode": "ApplyToTake",
    "CustomInfo": "custom123"
  }
}
```

## 请求包字段说明

| 字段                | 类型      | 属性 | 说明                                      |
|-------------------|---------|----|-----------------------------------------|
| RoomId            | String  | 必填 | 房间 ID                                   |
| RoomName          | String  | 选填 | 房间名字                                    |
| MaxMemberCount    | Integer | 选填 | 房间人数上限                                  |
| IsMessageDisabled | Bool    | 选填 | 房间全员禁言开关, 管理员和房主除外                      |
| CoverURL          | String  | 选填 | 房间封面                                    |
| Category          | Array   | 选填 | 房间分类标签, 数组长度上限为3                        |
| ActivityStatus    | Integer | 选填 | 房间自定义状态                                 |
| IsPublicVisible   | Bool    | 选填 | 房间开放开关, 主要用于获取房间列表使用                    |
| TakeSeatMode      | String  | 选填 | 麦位模式, 自由上麦 FreeToTake, 申请上麦 ApplyToTake |
| CustomInfo        | String  | 选填 | 自定义字段                                   |

## 应答包体示例



基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                                     |
|--------|------------------------------------------|
| 100001 | 服务器内部错误，请重试                              |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确                    |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了              |
| 100006 | 没有权限解散，例如操作者不是房主也不是root管理员或者解散的不是live房间等 |

# 获取房间信息

最近更新时间：2025-05-20 17:56:32

## 功能说明

App 管理员可以通过该接口获取房间信息。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/get_room_info?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                    | 说明                                                                                                                                                                           |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/get_room_info | 请求接口                                                                                                                                                                         |
| sdkappid                              | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                            | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                               | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                           | 请求格式固定值为 json                                                                                                                                                                |

### 最高调用频率

200次/秒。

### 请求包示例

## 基础形式

```
{
  "RoomId": "live-room"
}
```

## 请求包字段说明

| 字段     | 类型     | 属性 | 说明    |
|--------|--------|----|-------|
| RoomId | String | 必填 | 房间 ID |

## 应答包体示例

### 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "Response": { // 响应体
    "RoomInfo": {
      "RoomId": "rid-123",
      "RoomName": "rname-123",
      "RoomType": "Live",
      "Owner_Account": 144115233775727695,
      "CreateTime": 1693271354,
      "IsSeatEnabled": true,
      "TakeSeatMode": "ApplyToTake",
      "MaxSeatCount": 8,
      "MaxMemberCount": 300,
      "IsMessageDisabled": false,
      "CoverURL": "https://xxxx.png",
      "Category": [1, 2, 3],
      "ActivityStatus": 1,
      "ViewCount": 10,
      "IsPublicVisible": true
    }
  }
}
```

## 应答包字段说明

| 字段                | 类型      | 说明                                        |
|-------------------|---------|-------------------------------------------|
| ActionStatus      | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode         | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo         | String  | 错误信息                                      |
| RequestId         | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| RoomInfo          | Object  | 房间详情                                      |
| RoomId            | String  | 房间 ID                                     |
| RoomName          | String  | 房间名字                                      |
| RoomType          | String  | 房间类型：Live（直播房间）                           |
| Owner_Account     | String  | 房间群主 ID                                   |
| IsSeatEnabled     | Bool    | 麦位能力开关                                    |
| TakeSeatMode      | String  | 麦位模式，自由上麦 FreeToTake，申请上麦 ApplyToTake     |
| MaxSeatCount      | Integer | 麦位数量，受套餐包限制                               |
| MaxMemberCount    | Integer | 房间最大人数上限                                  |
| IsMessageDisabled | String  | 群内全员禁言开关                                  |
| CoverURL          | String  | 房间封面                                      |
| Category          | Array   | 直播间分类标签，数组大小上限为3                          |
| ActivityStatus    | Integer | 直播间活跃状态：用户自定义标记                           |
| IsPublicVisible   | Bool    | 房间是否公开可见, 用于获取直播间列表使用                     |
| CreateTime        | Integer | 房间创建时间                                    |
| ViewCount         | Integer | 用户进房总次数                                   |

### 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |

# 获取直播列表

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口获取直播列表。

## 接口调用说明

## 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/get_room_list?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                    | 说明                                                                                                                                                                           |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/get_room_list | 请求接口                                                                                                                                                                         |
| sdkappid                              | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                            | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                               | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                           | 请求格式固定值为 json                                                                                                                                                                |

## 最高调用频率

10次/秒。

## 请求包示例

## 基础形式

```
{
  "Next": "",
  "Count": 20
}
```

**注意:** 列表是按照ViewCount字段降序排序, 分页拉取

## 请求包字段说明

| 字段    | 类型      | 属性 | 说明                                                           |
|-------|---------|----|--------------------------------------------------------------|
| Next  | String  | 必填 | 分页请求的分页参数，第一次为空字符串，后续使用响应包体的 Next，当响应包体的 Next 为空字符串时候，表示拉取完毕 |
| Count | Integer | 必填 | 单次拉取的房间数量，默认为20，上限为20                                        |

## 应答包体示例

### 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "Response": {
    "Next": "1693271355-123123123123",
    "RoomList": [
      {
        "RoomName": "name",
        "RoomId": "room_id0",
        "Owner_Account": "bren",
        "CoverURL": "cover_url",
        "Category": [1, 2, 3],
        "CreateTime": 1693271354,
        "ActivityStatus": 1,
        "ViewCount": 1000
      },
      ...
    ]
  }
}
```

```

        "RoomId": "room_id19",
        "Owner_Account": "bren",
        "CoverURL": "cover_url",
        "Category": [1, 2, 3],
        "CreateTime": 1693271354,
        "ActivityStatus": 1,
        "ViewCount": 900
    }
}
}
}

```

## 应答包字段说明

| 字段             | 类型      | 说明                                         |
|----------------|---------|--------------------------------------------|
| ActionStatus   | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败                |
| ErrorCode      | Integer | 错误码，0表示成功，非0表示失败                           |
| ErrorInfo      | String  | 错误信息                                       |
| RequestId      | String  | 唯一请求 ID，每次请求都会返回, 定位问题时需要提供该次请求的 RequestId |
| RoomList       | Array   | 房间 ID 列表，默认按照 view count 降序排序              |
| RoomName       | String  | 房间名字                                       |
| RoomId         | String  | 房间 ID                                      |
| Owner_Account  | String  | 房间群主 ID                                    |
| CoverURL       | String  | 房间封面                                       |
| Category       | Array   | 直播间分类标签，数组大小上限为3                           |
| CreateTime     | Integer | 房间创建时间                                     |
| ActivityStatus | Integer | 直播间活跃状态：用户自定义标记                            |
| ViewCount      | Integer | 用户进房总次数                                    |
| Next           | String  | 当房间列表拉完后，返回空字符串，否则返回下次请求的 Next             |



## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                  |
|--------|-----------------------|
| 100001 | 服务器内部错误，请重试           |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确 |

# 解散房间

最近更新时间：2025-08-08 17:37:11

## 功能说明

应用管理员可以通过该接口解散房间。

**注意：**  
应用管理员账号可以参见文档 [应用管理员](#) 进行配置。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/destroy_room?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                   | 说明                                                                                                                                                                           |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                               | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/destroy_room | 解散房间请求接口                                                                                                                                                                     |
| sdkappid                             | 创建应用时即时通信 IM 控制台分配的 SDKAppID，可在即时通信 IM <a href="#">控制台</a> 的应用卡片中获取。                                                                                                         |
| identifier                           | 账号，必须为 App 管理员账号，更多详情请参见 <a href="#">应用管理员</a>                                                                                                                               |
| usersig                              | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                               | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                          | 请求格式，固定值为 json                                                                                                                                                               |

## 最高调用频率

200次/秒。

## 请求包示例

### 基础形式

```
{
  "RoomId": "live-room"
}
```

## 请求包字段说明

| 字段     | 类型     | 属性 | 说明    |
|--------|--------|----|-------|
| RoomId | String | 必填 | 直播间ID |

## 应答包体示例

### 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                                                                                     |
|--------------|---------|----------------------------------------------------------------------------------------|
| ActionStatus | String  | 请求处理的结果。 <ul style="list-style-type: none"><li>OK 表示处理成功。</li><li>FAIL 表示失败。</li></ul> |
| ErrorCode    | Integer | 错误码 <ul style="list-style-type: none"><li>0表示成功</li><li>非0表示失败</li></ul>               |
| ErrorInfo    | String  | 错误信息                                                                                   |

|           |        |                                           |
|-----------|--------|-------------------------------------------|
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
|-----------|--------|-------------------------------------------|

## 房间自动解散机制

Live 房间在以下三种情况会触发房间自动解散逻辑, 支持 sdkappid 维度的配置，开启条件和时间的调整配置请联系商务或者 [提交工单](#) 进行变更。

1. 当有用户进入过房间之后，房间空无一人并且持续一段时间内(默认30分钟)没人进房，就会触发自动解散机制，配置 [机器人](#) 的房间不会自动解散。2025年05月20日之前的 Live 客户维持原来的6小时解散机制，该时间支持调整，配置区间为[10分钟，24小时]，**该机制不支持关闭**。
2. 房主调用接口离开房间并且持续一段时间内(默认10分钟)没有重新进房，就会触发自动解散机制，如果房主是 [机器人](#)，则不会触发该机制，该机制**默认关闭**，该时间支持调整，配置区间为[2分钟，12小时]。
3. 房主超过心跳保活时间掉线后，并且持续一段时间内(默认10分钟)没有恢复心跳进房，就会触发自动解散机制，如果房主是 [机器人](#)，则不会触发该机制，该机制**默认关闭**，该时间支持调整，配置区间为[2分钟，12小时]。

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                                         |
|--------|----------------------------------------------|
| 100001 | 服务器内部错误，请重试                                  |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确                        |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了                  |
| 100006 | 没有权限解散，例如操作者不是房主也不是 root 管理员或者解散的不是 live 房间等 |

# 获取房间自定义信息

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口设置房间自定义信息。

## 接口调用说明

## 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/get_room_metadata?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                        | 说明                                                                                                                                                                           |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                    | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/get_room_metadata | 请求接口                                                                                                                                                                         |
| sdkappid                                  | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                                | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                                   | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                    | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                               | 请求格式固定值为 json                                                                                                                                                                |

## 最高调用频率

200次/秒。

## 请求包示例

## 基础形式

```
{
  "RoomId": "live-room",
  "Keys": ["key1", "key2"] //如果为空数组表示获取该房间的所有的自定义信息，否则就是
  获取指定Key列表的自定义信息
}
```

## 请求包字段说明

| 字段     | 类型     | 属性 | 说明                               |
|--------|--------|----|----------------------------------|
| RoomId | String | 必填 | 房间 ID                            |
| Keys   | Array  | 必填 | 指定的 Key 列表，如果为空，则表示获取房间内所有的自定义信息 |

## 应答包体示例

### 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "Response": {
    "Metadata": [
      {
        "Key": "key1",
        "Value": "value1"
      },
      {
        "Key": "key2",
        "Value": "value2"
      }
    ]
  }
}
```

## 应答包字段说明

| 字段 | 类型 | 说明 |
|----|----|----|
|----|----|----|

|              |         |                                           |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| Response     | Object  | 房间自定义信息                                   |
| Metadata     | Array   | 返回的有效的房间自定义信息                             |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                             |
|--------|----------------------------------|
| 100001 | 服务器内部错误，请重试                      |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确            |
| 100003 | 房间 ID 已存在，请选择其他房间 ID             |
| 100006 | 无权限操作房间自定义信息                     |
| 100007 | 无付费信息，需在控制台购买套餐包                 |
| 100500 | 房间自定义信息key数量超过上限10               |
| 100501 | 房间自定义信息单个key对应的value字节数超过上限2KB   |
| 100502 | 房间自定义信息所有key对应的value字节数总和超过上限1KB |
| 100503 | 删除自定义信息时候指定的key没有一个有效的           |

# 设置房间自定义信息

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口设置房间自定义信息。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/set_room_metadata?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype  
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                        | 说明                                                                                                                                                                        |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                    | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国：console.tim.qq.com</li><li>新加坡：adminapisgp.im.qcloud.com</li><li>美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/set_room_metadata | 请求接口                                                                                                                                                                      |
| sdkappid                                  | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                              |
| identifier                                | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                             |
| usersig                                   | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                         |
| random                                    | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                         |
| contenttype                               | 请求格式固定值为 json                                                                                                                                                             |

### 最高调用频率

200次/秒。

### 请求包示例



## 基础形式

```
{
  "RoomId": "live-room",
  "Metadata": [ //key数量不超过10, key 的长度不超过50字节, 单个value 字节数上限
    为 2K, 所有value总字节数上限为16K
    {
      "Key": "key1", //当key存在时候则是更新, 否则就是新增
      "Value": "value1"
    },
    {
      "Key": "key2",
      "Value": "value2"
    }
  ]
}
```

## 请求包字段说明

| 字段       | 类型     | 属性 | 说明                                                                             |
|----------|--------|----|--------------------------------------------------------------------------------|
| RoomId   | String | 必填 | 房间 ID                                                                          |
| Metadata | Array  | 必填 | 房间自定义信息列表, key 数量不超过10, key 的长度不超过50字节, 单个 value 字节数上限为2K, 所有 value 总字节数上限为16K |

## 应答包体示例

### 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-0-Seq-52720"
}
```

## 应答包字段说明

| 字段           | 类型     | 说明                            |
|--------------|--------|-------------------------------|
| ActionStatus | String | 请求处理的结果, OK 表示处理成功, FAIL 表示失败 |

|           |         |                                           |
|-----------|---------|-------------------------------------------|
| ErrorCode | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo | String  | 错误信息                                      |
| RequestId | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                                  |
|--------|---------------------------------------|
| 100001 | 服务器内部错误，请重试                           |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确                 |
| 100003 | 房间 ID 已存在，请选择其他房间 ID                  |
| 100006 | 无权限操作房间自定义信息                          |
| 100007 | 无付费信息，需在控制台购买套餐包                      |
| 100500 | 房间自定义信息 key 数量超过上限10                  |
| 100501 | 房间自定义信息单个 key 对应的 value 字节数超过上限2KB    |
| 100502 | 房间自定义信息所有 key 对应的 value 字节数总和超过上限16KB |
| 100503 | 删除自定义信息时候指定的 key 没有一个有效的              |

# 删除房间自定义信息

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口删除房间内指定的间自定义信息。

## 接口调用说明

## 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/del_room_metadata?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                        | 说明                                                                                                                                                                           |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                    | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/del_room_metadata | 请求接口                                                                                                                                                                         |
| sdkappid                                  | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                                | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                                   | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                    | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                               | 请求格式固定值为 json                                                                                                                                                                |

## 最高调用频率

200次/秒。

## 请求包示例

## 基础形式

```
{
  "RoomId": "live-room",
  "Keys": ["key1", "key2"]
}
```

## 请求包字段说明

| 字段     | 类型     | 属性 | 说明                  |
|--------|--------|----|---------------------|
| RoomId | String | 必填 | 房间 ID               |
| Keys   | Array  | 必填 | 房间自定义信息指定删除的 Key 列表 |

## 应答包体示例

### 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-0-Seq-52720"
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                                  |
|--------|---------------------------------------|
| 100001 | 服务器内部错误，请重试                           |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确                 |
| 100003 | 房间 ID 已存在，请选择其他房间 ID                  |
| 100006 | 无权限操作房间自定义信息                          |
| 100007 | 无付费信息，需在控制台购买套餐包                      |
| 100500 | 房间自定义信息 key 数量超过上限10                  |
| 100501 | 房间自定义信息单个 key 对应的 value 字节数超过上限2KB    |
| 100502 | 房间自定义信息所有 key 对应的 value 字节数总和超过上限16KB |
| 100503 | 删除自定义信息时候指定的key没有一个有效的                |

# 房间成员管理

## 成员禁言

最近更新时间：2025-04-03 17:08:43

### 功能说明

App 管理员可以通过该接口对成员进行房间内禁言和取消禁言。

### 接口调用说明

#### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/mute_member?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                  | 说明                                                                                                                                                                              |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                              | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国：console.tim.qq.com</li><li>● 新加坡：adminapisgp.im.qcloud.com</li><li>● 美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/mute_member | 请求接口                                                                                                                                                                            |
| sdkappid                            | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                    |
| identifier                          | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                   |
| usersig                             | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                               |
| random                              | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                               |
| contenttype                         | 请求格式固定值为 json                                                                                                                                                                   |

### 最高调用频率

200次/秒。

请求包示例

基础形式

```
{
  "RoomId": "test",
  "MemberList_Account": [
    "user1",
    "user2"
  ],
  "MuteTime": 3600
}
```

请求包字段说明

| 字段                 | 类型      | 属性 | 说明               |
|--------------------|---------|----|------------------|
| RoomId             | String  | 必填 | 房间 ID            |
| MemberList_Account | Array   | 必填 | 禁言的成员列表          |
| MuteTime           | integer | 必填 | 禁言时间，如果是0，表示取消禁言 |

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-0-Seq-52720"
}
```

应答包字段说明

| 字段           | 类型      | 说明                          |
|--------------|---------|-----------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败            |
| ErrorInfo    | String  | 错误信息                        |

|           |        |                                           |
|-----------|--------|-------------------------------------------|
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
|-----------|--------|-------------------------------------------|

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码（TUILiveKit）](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |



# 获取禁言列表

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口针对成员进行禁言和取消禁言。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/get_muted_member_list?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                            | 说明                                                                                                                                                                        |
|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                        | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国：console.tim.qq.com</li><li>新加坡：adminapisgp.im.qcloud.com</li><li>美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/get_muted_member_list | 请求接口                                                                                                                                                                      |
| sdkappid                                      | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                              |
| identifier                                    | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                             |
| usersig                                       | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                         |
| random                                        | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                         |
| contenttype                                   | 请求格式固定值为 json                                                                                                                                                             |

### 最高调用频率

200次/秒。

### 请求包示例

基础形式

```
{
  "RoomId": "test"
}
```

请求包字段说明

| 字段     | 类型     | 属性 | 说明    |
|--------|--------|----|-------|
| RoomId | String | 必填 | 房间 ID |

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-9a9fe01d96164827b49dacb6b3cdadba-O-Seq-126318",
  "Response": {
    "MutedAccountList": [
      {
        "Member_Account": "user1",
        "MutedUntil": 1719293107
      },
      {
        "Member_Account": "brennanli",
        "MutedUntil": 1719293107
      }
    ]
  }
}
```

应答包字段说明

| 字段           | 类型      | 说明                          |
|--------------|---------|-----------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败            |

|                  |         |                                           |
|------------------|---------|-------------------------------------------|
| ErrorInfo        | String  | 错误信息                                      |
| RequestId        | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| MutedAccountList | Array   | 禁言列表                                      |
| Member_Account   | String  | 禁言用户 ID                                   |
| MutedUntil       | Integer | 禁言截止时间                                    |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |

# 成员封禁

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口封禁成员，被封禁的用户在封禁有效期内无法再进房间。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/ban_member?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                 | 说明                                                                                                                                                                           |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                             | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/ban_member | 请求接口                                                                                                                                                                         |
| sdkappid                           | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                         | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                            | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                             | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                        | 请求格式固定值为 json                                                                                                                                                                |

### 最高调用频率

200次/秒。

### 请求包示例

基础形式

```
{
  "RoomId": "test",
  "MemberList_Account": [
    "user1",
    "user2"
  ],
  "Reason": "",
  "Duration": 3600
}
```

请求包字段说明

| 字段                 | 类型      | 属性 | 说明               |
|--------------------|---------|----|------------------|
| RoomId             | String  | 必填 | 房间 ID            |
| MemberList_Account | Array   | 必填 | 被封禁的成员列表，单次最多20个 |
| Reason             | String  | 选填 | 封禁原因             |
| Duration           | Integer | 必填 | 封禁时间             |

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

| 字段           | 类型      | 说明                          |
|--------------|---------|-----------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败            |

|           |        |                                           |
|-----------|--------|-------------------------------------------|
| ErrorInfo | String | 错误信息                                      |
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |
| 100006 | 没有权限转移房主                    |
| 100016 | 新的房主被封禁，请先解禁                |

# 解除成员封禁

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口解除成员封禁。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/unban_member?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype  
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                   | 说明                                                                                                                                                                        |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                               | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国：console.tim.qq.com</li><li>新加坡：adminapisgp.im.qcloud.com</li><li>美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/unban_member | 请求接口                                                                                                                                                                      |
| sdkappid                             | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                              |
| identifier                           | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                             |
| usersig                              | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                         |
| random                               | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                         |
| contenttype                          | 请求格式固定值为 json                                                                                                                                                             |

### 最高调用频率

200次/秒。

### 请求包示例

基础形式

```
{
  "RoomId": "test",
  "MemberList_Account": [
    "user1",
    "user2"
  ]
}
```

请求包字段说明

| 字段                 | 类型     | 属性 | 说明                |
|--------------------|--------|----|-------------------|
| RoomId             | String | 必填 | 房间 ID             |
| MemberList_Account | Array  | 必填 | 解除封禁的成员列表，单次最多20个 |

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

错误码说明



除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |

# 修改管理员

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口进行设置和取消房间管理员。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/modify_admin?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                       | 说明                                                                                                                                                                        |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                   | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国：console.tim.qq.com</li><li>新加坡：adminapisgp.im.qcloud.com</li><li>美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/mod<br>ify_admin | 请求接口                                                                                                                                                                      |
| sdkappid                                 | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                              |
| identifier                               | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                             |
| usersig                                  | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                         |
| random                                   | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                         |
| contenttype                              | 请求格式固定值为 json                                                                                                                                                             |

### 最高调用频率

200次/秒。

### 请求包示例

基础形式

```
{
  "RoomId": "test",
  "Admin_Account": [
    "user1",
    "user2"
  ],
  "OpType": "add"
}
```

请求包字段说明

| 字段            | 类型     | 属性 | 说明                              |
|---------------|--------|----|---------------------------------|
| RoomId        | String | 必填 | 房间 ID                           |
| Admin_Account | Array  | 必填 | 修改的管理员列表，最大支持 5 个管理员            |
| OpType        | String | 必填 | 操作类型，add 表示添加管理员，delete 表示删除管理员 |

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

应答包字段说明

| 字段           | 类型      | 说明                          |
|--------------|---------|-----------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败            |
| ErrorInfo    | String  | 错误信息                        |

|           |        |                                           |
|-----------|--------|-------------------------------------------|
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
|-----------|--------|-------------------------------------------|

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |
| 100020 | 房间管理员超过上限, 最大上限为5           |

# 获取封禁列表

最近更新时间：2025-05-20 17:56:32

## 功能说明

App 管理员可以通过该接口获取封禁成员列表。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/get_banned_member_list?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                             | 说明                                                                                                                                                                           |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                         | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/get_banned_member_list | 请求接口                                                                                                                                                                         |
| sdkappid                                       | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                                     | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                                        | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                         | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                                    | 请求格式固定值为 json                                                                                                                                                                |

## 最高调用频率

200次/秒。

## 请求包示例

## 基础形式

```
{
  "RoomId": "test",
  "Offset": 0,
  "Limit": 10
}
```

## 请求包字段说明

| 字段     | 类型      | 属性 | 说明                                  |
|--------|---------|----|-------------------------------------|
| RoomId | String  | 必填 | 房间 ID                               |
| Offset | Integer | 选填 | 偏移量，初次拉取从0开始，后续请求以响应包 NextOffset 为准 |
| Limit  | Integer | 选填 | 单次分页拉取的数量，最大为100                    |

## 应答包体示例

### 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-138c94122acc4ca485c8c7bf270686ac-0-Seq-135511",
  "Response": { // 响应体
    "BannedAccountList": [
      {
        "Member_Account": "user2",
        "BannedUntil": 1719300627
      },
      {
        "Member_Account": "user1",
        "BannedUntil": 1719300627
      }
    ],
    "NextOffset": 0 // 分页标识
  }
}
```

## 应答包字段说明

| 字段                | 类型      | 说明                                        |
|-------------------|---------|-------------------------------------------|
| ActionStatus      | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode         | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo         | String  | 错误信息                                      |
| RequestId         | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| BannedAccountList | Array   | 封禁成员列表                                    |
| Member_Account    | String  | 封禁成员账号 ID                                 |
| BannedUntil       | Integer | 封禁截止时间                                    |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |

# 获取管理员列表

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口获取房间管理员列表。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/get_admin_list?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                     | 说明                                                                                                                                                                           |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                 | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/get_admin_list | 请求接口                                                                                                                                                                         |
| sdkappid                               | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                             | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                                | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                 | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                            | 请求格式固定值为 json                                                                                                                                                                |

### 最高调用频率

200次/秒。

### 请求包示例



基础形式

```
{
  "RoomId": "test"
}
```

请求包字段说明

| 字段     | 类型     | 属性 | 说明    |
|--------|--------|----|-------|
| RoomId | String | 必填 | 房间 ID |

应答包体示例

基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720",
  "Response": { // 响应体
    "Admin_Account": [
      "user2",
      "user1"
    ]
  }
}
```

应答包字段说明

| 字段            | 类型      | 说明                                        |
|---------------|---------|-------------------------------------------|
| ActionStatus  | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode     | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo     | String  | 错误信息                                      |
| RequestId     | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| Admin_Account | Array   | 管理员列表                                     |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的

ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                        |
|--------|-----------------------------|
| 100001 | 服务器内部错误，请重试                 |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确       |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了 |

# 获取房间成员列表

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口获取房间成员列表。

## 接口调用说明

## 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/get_member_list?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

 **注意：**

- 在线成员列表总体更新粒度为40s。
- 当房间超过1000人时，接口仅返回1000人。

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                      | 说明                                                                                                                                                                                 |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                  | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>• 中国： console.tim.qq.com</li><li>• 新加坡： adminapisgp.im.qcloud.com</li><li>• 美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/get_member_list | 请求接口                                                                                                                                                                               |
| sdkappid                                | 创建应用时分配的 SDKAppID                                                                                                                                                                  |
| identifier                              | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                      |
| usersig                                 | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                                  |
| random                                  | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                                  |
| contenttype                             | 请求格式固定值为 json                                                                                                                                                                      |

## 最高调用频率

200次/秒。

## 请求包示例

### 基础形式

```
{
  "RoomId": "room_id"
  "Next": "" // 首次为空字符串，下次请求为之前响应里的Next，直到为空字符串表示已
经获取完毕
}
```

## 请求包字段说明

| 字段     | 类型     | 属性 | 说明     |
|--------|--------|----|--------|
| RoomId | String | 必填 | 房间 ID  |
| Next   | String | 必填 | 分页获取参数 |

## 应答包体示例

### 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "requestId-123-456",
  "Response" : {
    "Next": "", //返回空表示所有成员获取完毕
    "MemberCount": 2, //房间总人数
    "MemberList": [
      {
        "Member_Account" : "brennan1"
      },
      {
        "Member_Account" : "tandy"
      },
    ],
  ]
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| MemberList   | Array   | 房间成员列表                                    |
| Next         | String  | 下次请求参数里的Next值                             |
| MemberCount  | Integer | 房间总人数                                     |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码（TUILiveKit）](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                            |
|--------|---------------------------------|
| 100001 | 系统错误，重试                         |
| 100004 | 房间不存在，原因可能是房间没被创建过，或者已经被解散了     |
| 100006 | 表示此房间不是 live 房间，无权限使用 live 接口查询 |

# 连麦管理

## 获取麦位列表

最近更新时间：2025-04-03 17:08:43

### 功能说明

App 管理员可以通过该接口获取麦位列表。

### 接口调用说明

#### 请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/get_seat_list?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                    | 说明                                                |
|---------------------------------------|---------------------------------------------------|
| v4/room_engine_http_mic/get_seat_list | 请求接口                                              |
| sdkappid                              | 创建应用时即时通信 IM 控制台分配的 SDKAppID                      |
| identifier                            | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>     |
| usersig                               | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a> |
| random                                | 请输入随机的32位无符号整数，取值范围0 – 4294967295                 |
| contenttype                           | 请求格式固定值为 <code>json</code>                        |

### 最高调用频率

200次/秒。

### 请求包示例

#### 基础形式

```
{
```

```
"RoomId": "room-test"
}
```

请求包字段说明

| 字段     | 类型     | 属性 | 说明    |
|--------|--------|----|-------|
| RoomId | String | 必填 | 房间 ID |

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717",
  "Response": {
    "SeatList": [
      {
        "Index": 0,
        "Member_Account": "user2",
        "IsTakenDisabled": false,
        "IsVideoDisabled": false,
        "IsAudioDisabled": false
      },
      {
        "Index": 1,
        "IsTakenDisabled": false,
        "IsVideoDisabled": false,
        "IsAudioDisabled": false
      }
    ]
  }
}
```

应答包字段说明

| 字段           | 类型     | 说明                          |
|--------------|--------|-----------------------------|
| ActionStatus | String | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |

|                 |         |                                           |
|-----------------|---------|-------------------------------------------|
| ErrorCode       | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo       | String  | 错误信息                                      |
| RequestId       | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| SeatList        | Array   | 麦位列表                                      |
| Index           | Integer | 麦位号                                       |
| Member_Account  | String  | 麦位上的用户 ID，空表示麦上没有用户                       |
| IsTakenDisabled | Bool    | 麦位锁定                                      |
| IsVideoDisabled | Bool    | 禁止麦位视频流                                   |
| IsAudioDisabled | Bool    | 禁止麦位音频流                                   |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                    |
|--------|-------------------------|
| 100001 | 服务器内部错误，请重试             |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确   |
| 100004 | 房间不存在，或者曾经存在过，但是目前已经被解散 |



# 用户上麦

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口让指定成员上麦。

## 接口调用说明

### 请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/pick_user_on_seat?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                        | 说明                                                |
|-------------------------------------------|---------------------------------------------------|
| v4/room_engine_http_mic/pick_user_on_seat | 请求接口                                              |
| sdkappid                                  | 创建应用时即时通信 IM 控制台分配的 SDKAppID                      |
| identifier                                | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>     |
| usersig                                   | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a> |
| random                                    | 请输入随机的32位无符号整数，取值范围0 – 4294967295                 |
| contenttype                               | 请求格式固定值为 <code>json</code>                        |

### 最高调用频率

200次/秒。

### 请求包示例

#### 基础形式

```
{
  "RoomId": "room-test",
  "Member_Account": "user2",
```

```
"Index":1
}
```

请求包字段说明

| 字段             | 类型      | 属性 | 说明              |
|----------------|---------|----|-----------------|
| RoomId         | String  | 必填 | 房间 ID           |
| Index          | Integer | 必填 | 麦位号             |
| Member_Account | String  | 必填 | 上麦用户 ID，需要为房间成员 |

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-0-Seq-57717"
}
```

应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                    |
|--------|-------------------------|
| 100001 | 服务器内部错误，请重试             |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确   |
| 100004 | 房间不存在，或者曾经存在过，但是目前已经被解散 |
| 100005 | 上麦用户非房间成员               |
| 100200 | 麦位已锁定，可以尝试换一个麦位         |
| 100202 | 已经处于排麦状态                |
| 100203 | 已经处于麦上状态                |
| 100205 | 麦位已满                    |
| 100210 | 已有用户在麦位上                |
| 100211 | 该房间不支持连麦                |

## 可能触发的回调

- [麦位列表变更之后回调](#)

# 用户下麦

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口让指定成员下麦。

## 接口调用说明

### 请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/kick_user_off_seat?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                         | 说明                                                |
|--------------------------------------------|---------------------------------------------------|
| v4/room_engine_http_mic/kick_user_off_seat | 请求接口                                              |
| sdkappid                                   | 创建应用时即时通信 IM 控制台分配的 SDKAppID                      |
| identifier                                 | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>     |
| usersig                                    | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a> |
| random                                     | 请输入随机的32位无符号整数，取值范围0 – 4294967295                 |
| contenttype                                | 请求格式固定值为 <code>json</code>                        |

### 最高调用频率

200次/秒。

### 请求包示例

#### 基础形式

```
{
  "RoomId": "room-test",
  "Member_Account": "user2",
```

```
"Data": "time over"
}
```

请求包字段说明

| 字段             | 类型     | 属性 | 说明                   |
|----------------|--------|----|----------------------|
| RoomId         | String | 必填 | 房间 ID                |
| Member_Account | String | 必填 | 麦上用户 ID              |
| Data           | String | 必填 | 自定义信息，该信息踢人下麦下行通知时下发 |

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-0-Seq-57717"
}
```

应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                    |
|--------|-------------------------|
| 100001 | 服务器内部错误，请重试             |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确   |
| 100004 | 房间不存在，或者曾经存在过，但是目前已经被解散 |

## 可能触发的回调

- [麦位列表变更之后回调](#)

# 锁定麦位

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口锁定麦位，麦位锁定后，用户无法再上此麦位。锁定麦位时，如果麦上有人，会先将用户下麦，再锁定。

## 接口调用说明

### 请求 URL 示例

```
https://console.tim.qq.com/v4/room_engine_http_mic/lock_seat?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                | 说明                                                |
|-----------------------------------|---------------------------------------------------|
| v4/room_engine_http_mic/lock_seat | 请求接口                                              |
| sdkappid                          | 创建应用时即时通信 IM 控制台分配的 SDKAppID                      |
| identifier                        | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>     |
| usersig                           | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a> |
| random                            | 请输入随机的32位无符号整数，取值范围0 – 4294967295                 |
| contenttype                       | 请求格式固定值为 <code>json</code>                        |

### 最高调用频率

200次/秒。

### 请求包示例

#### 基础形式

```
{
```

```
"RoomId": "room-test",
"Index": 2,
"ForbidTaken": true,
"ForbidVideo": true,
"ForbidAudio": true
}
```

## 请求包字段说明

| 字段          | 类型      | 属性 | 说明    |
|-------------|---------|----|-------|
| RoomId      | String  | 必填 | 房间 ID |
| Index       | Integer | 必填 | 麦位号   |
| ForbidTaken | Bool    | 选填 | 锁定麦位  |
| ForbidVideo | Bool    | 选填 | 禁止视频流 |
| ForbidAudio | Bool    | 选填 | 禁止音频流 |

## 应答包体示例

### 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-7c1680be52734bdc8d6de398ab9505e7-O-Seq-57717"
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                          |
|--------------|---------|-----------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败            |



|           |        |                                           |
|-----------|--------|-------------------------------------------|
| ErrorInfo | String | 错误信息                                      |
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                    |
|--------|-------------------------|
| 100001 | 服务器内部错误，请重试             |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确   |
| 100004 | 房间不存在，或者曾经存在过，但是目前已经被解散 |
| 100211 | 该房间不支持连麦                |

## 可能触发的回调

- [麦位列表变更之后回调](#)

# PK 相关

## 发起 PK

最近更新时间：2025-05-20 17:56:32

### 功能说明

App 管理员可以通过该接口发起 PK。

### 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_PK/create_battle?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                   | 说明                                                                                                                                                                                 |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                               | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国： console.tim.qq.com</li><li>● 新加坡： adminapisgp.im.qcloud.com</li><li>● 美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_pk/create_battle | 请求接口                                                                                                                                                                               |
| sdkappid                             | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                       |
| identifier                           | 必须为 App 管理员账号，更多详情请参见 <a href="#">App管理员</a>                                                                                                                                       |
| usersig                              | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成UserSig</a>                                                                                                                                   |
| random                               | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                                  |
| contenttype                          | 请求格式固定值为 json                                                                                                                                                                      |

### 最高调用频率

200次/秒。

请求包示例

基础形式

```
{
  "FromRoomId": "pk-1",    //pk的主叫房间id
  "ToRoomIdList": ["pk-2", "pk-3"], //pk的被叫房间id列表
  "Duration": 60000, //pk时长
  "ExtensionInfo": "extension pk"
}
```

请求包字段说明

| 字段            | 类型      | 属性 | 说明            |
|---------------|---------|----|---------------|
| FromRoomId    | String  | 必填 | PK 主叫房间 ID    |
| ToRoomIdList  | String  | 必填 | PK 被叫房间 ID 列表 |
| Duration      | Integer | 必填 | PK 时长         |
| ExtensionInfo | String  | 选填 | PK 信息扩展字段     |

 **注意：**

发起 PK 时候，业务侧需要确认端上参与 PK 的房间处于连线中。发起 PK 的时候会通知参与 PK 的房间房主。

应答包体示例

基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "requestId-123-456",
  "Response": {
    "BattleId": "battleid1",
    "Result": [
      {
        "RoomId": "pk-2",
        "ResultCode": 0,
        // 0 表示发起成功，
      }
    ]
  }
}
```

```
// 3 表示已经处于其他PK中,
// 5 系统错误
"ResultMessage": "",
},
{
  "RoomId": "pk-3",
  "ResultCode": 0,
  "ResultMessage": "",
}
],
}
```

## 应答包字段说明

| 字段            | 类型      | 说明                                        |
|---------------|---------|-------------------------------------------|
| ErrorCode     | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo     | String  | 错误信息                                      |
| ActionStatus  | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| RequestId     | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| BattleId      | String  | 此次 PK 的 ID                                |
| Result        | Array   | 被叫的 PK 结果                                 |
| RoomId        | String  | 房间 ID                                     |
| ResultCode    | Integer | 结果，0表示成功，3表示正处于 pk 中，5表示系统错误              |
| ResultMessage | String  | 结果信息                                      |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码 | 含义说明 |
|-----|------|
|-----|------|

|        |                       |
|--------|-----------------------|
| 100001 | 服务器内部错误，请重试           |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确 |
| 100003 | 房间 ID 已存在，请选择其他房间 ID  |
| 100412 | 被叫房间都处于 PK 中，请稍后再试    |
| 100415 | 主叫方处于 PK 中，请稍后再试      |

# 修改 PK 分数

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口修改 pk 里各房间分数。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_pk/set_battle_score?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明。

| 参数                                      | 说明                                                                                                                                                                        |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                  | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国：console.tim.qq.com</li><li>新加坡：adminapisgp.im.qcloud.com</li><li>美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_pk/set_battle_score | 请求接口                                                                                                                                                                      |
| sdkappid                                | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                              |
| identifier                              | 必须为 App 管理员账号，更多详情请参见 <a href="#">App管理员</a>                                                                                                                              |
| usersig                                 | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成UserSig</a>                                                                                                                          |
| random                                  | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                         |
| contenttype                             | 请求格式固定值为 json                                                                                                                                                             |

### 最高调用频率

200次/秒。

### 请求包示例

## 基础形式

```
{
  "BattleId": "HuYM036AMk2JJJeSUgnv5H0KDJpc+HXfqTAtGuYWzRPs=",
  "OpType": 1, // 1表示累加, 2表示重置
  "ScoreList": [
    {
      "RoomId": "pk-live1",
      "Score": 100
    },
    {
      "RoomId": "pk-live2",
      "Score": 100
    }
  ]
}
```

## 请求包字段说明

| 字段        | 类型      | 属性 | 说明                                 |
|-----------|---------|----|------------------------------------|
| BattleId  | String  | 必填 | pk 场次对应的 ID，可通过获取 pk 详情获取到         |
| OpType    | String  | 必填 | 设置类型，1为累加，2为覆盖设置                   |
| ScoreList | Array   | 必填 | 设置分数列表，如果请求里的房间不处于该 pk 中，则该列表忽略此房间 |
| RoomId    | String  | 必填 | 房间 ID                              |
| Score     | Integer | 必填 | 房间分数                               |

## 应答包体示例

### 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-e7625183568d4058bd48c9924d4b1ed7-O-Seq-125382",
  "Response": {
    "Result": [
```

```
{
  "RoomId": "pk-3",
  "ResultCode": 0
},
{
  "RoomId": "pk-4",
  "ResultCode": 0
}
]
```

## 应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| Result       | Array   | 操作结果                                      |
| RoomId       | String  | 房间 id                                     |
| ResultCode   | Integer | 0表示成功，1表示房间不存在                            |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                     |
|--------|--------------------------|
| 100001 | 服务器内部错误，请重试              |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确    |
| 100006 | 当前 pk 尚未开始或者已经结束，无权再设置分数 |



# 获取 PK 信息

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口获取 pk 详情。

## 接口调用说明

## 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_pk/get_battle_info?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                     | 说明                                                                                                                                                                                 |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                 | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国： console.tim.qq.com</li><li>● 新加坡： adminapisgp.im.qcloud.com</li><li>● 美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_pk/get_battle_info | 请求接口                                                                                                                                                                               |
| sdkappid                               | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                       |
| identifier                             | 必须为 App 管理员账号，更多详情请参见 <a href="#">App管理员</a>                                                                                                                                       |
| usersig                                | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成UserSig</a>                                                                                                                                   |
| random                                 | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                                  |
| contenttype                            | 请求格式固定值为 json                                                                                                                                                                      |

## 最高调用频率

200次/秒。

## 请求包示例

## 基础形式

```
{
  "BattleId": "HuYM036AMk2JJeSUgnv5H0KDJpc+HXfqTAtGuYWzRPs=", //业务后台
  //创建pk时候可以保存好该BattleId。
  "RoomId": "room_pk1" //当业务侧不知道具体的BattleId时候，可以填写RoomId，这
  //样就可以获取该RoomId正在进行的BattleId信息
}
```

## 请求包字段说明

| 字段       | 类型     | 属性 | 说明                                                     |
|----------|--------|----|--------------------------------------------------------|
| BattleId | String | 必填 | pk 场次的 ID                                              |
| RoomId   | String | 选填 | 房间 ID，当不知道 BattleId 时候，可以填写 RoomId，响应包返回该房间正在进行的 PK 详情 |

## 应答包体示例

### 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-e1ffffbc751dc44bc95ec960d1fdfbefe-O-Seq-126652",
  "Response": {
    "BattleId": "HuYM036AMk2JJeSUgnv5H0KDJpc+HXfqTAtGuYWzRPs=",
    "Duration": 120000,
    "StartTime": 1739508625,
    "EndTime": 1739508746,
    "Status": 3,
    "FromRoomInfo": {
      "RoomId": "pk-3",
      "Score": 100,
      "Owner_Account": "brennanli1"
    },
    "ToRoomList": [
      {
        "RoomId": "pk-4",
        "Score": 100,
        "Owner_Account": "brennanli2"
      }
    ]
  }
}
```

```
    }
  ]
}
}
```

应答包字段说明

| 字段            | 类型      | 说明                                        |
|---------------|---------|-------------------------------------------|
| ErrorCode     | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo     | String  | 错误信息                                      |
| ActionStatus  | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| RequestId     | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| BattleId      | String  | pk ID                                     |
| Duration      | Integer | pk 时长                                     |
| StartTime     | Integer | pk 正式开始时间                                 |
| EndTime       | Integer | pk 正式结束时间                                 |
| Status        | Integer | pk 状态，1表示未开始，2表示进行中，3表示已结束，4表示 pk 未开始就结束  |
| FromRoomInfo  | Object  | pk 主叫房间信息                                 |
| ToRoomList    | Array   | pk 被叫房间信息                                 |
| RoomId        | String  | 房间 ID                                     |
| Score         | Integer | pk 中的房间分数                                 |
| Owner_Account | String  | pk 中的房间对应的房主账号 ID                         |

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                   |
|--------|------------------------|
| 100001 | 服务器内部错误，请重试            |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确  |
| 100411 | 当前pk不存在，或者结束时间过长信息已被清理 |

# 云端搜索

## 房间成员搜索

最近更新时间：2025-04-03 17:08:43

### 功能说明

App 管理员可以通过该接口在房间里模糊搜索成员。

### 请求 URL 示例

```
https://xxxxxx/v4/live_engine_http_srv/search_room_member?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                                         | 说明                                                                                                                                                                           |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                                     | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| v4/live_engine_http_srv/search_room_member | 请求接口                                                                                                                                                                         |
| sdkappid                                   | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                                 | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                                    | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                                     | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                                | 请求格式固定值为 json                                                                                                                                                                |

### 最高调用频率

2次/秒。

### 请求包示例

## 基础形式

```
{
  "RoomId": "search_test", // 房间id
  "Keyword": "brennanli", // 关键字搜索，用户的账号昵称或者账号id，账号昵称支持模糊搜索，账号id支持精确匹配
  "Next": "" // 用于分页查询的参数，用于指定下一页数据的起始位置，首次请求时，Next 字段为空，每次请求后，服务器返回响应中的 Next 字段值。客户端需将该值作为下一次请求的 Next 参数，直到 Next 为空才停止
}
```

## 请求包字段说明

| 字段      | 类型     | 属性 | 说明                                |
|---------|--------|----|-----------------------------------|
| RoomId  | String | 必填 | 房间 ID                             |
| Keyword | String | 必填 | 关键字匹配，可填写账号昵称或者账号 ID，其中账号昵称支持模糊搜索 |
| Next    | String | 必填 | 用于分页查询的参数，用于指定下一页数据的起始位置          |

### ⚠ 注意：

该接口能力支持版本为：体验版 + LiveKit 大规模直播版，当前需要提单联系平台手动打开。

## 应答包体示例

### 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-1ea25224fab7403f939986c2f7266ce2-0-Seq-1645192760",
  "Response": {
    "Next": "",
    "MemberList": [
      {
        "Member_Account": "brennanli1",
        "NickName": "brennanliqbrennanliqbrennanliqbrennanliqbrennanliqbrennanliq"
      }
    ]
  }
}
```

```
{
  "Member_Account": "brennanli",
  "NickName": "brennanli_nick"
}
```

应答包字段说明

| 字段             | 类型      | 说明                                         |
|----------------|---------|--------------------------------------------|
| ErrorCode      | Integer | 错误码，0表示成功，非0表示失败                           |
| ErrorInfo      | String  | 错误信息                                       |
| ActionStatus   | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败                |
| RequestId      | String  | 唯一请求 ID，每次请求都会返回, 定位问题时需要提供该次请求的 RequestId |
| Next           | String  | 分页列表的参数，为空时候表示结束                           |
| MemberList     | Array   | 搜索到的成员信息列表                                 |
| Member_Account | String  | 成员的账号 ID                                   |
| NickName       | String  | 成员的账号昵称                                    |

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                  |
|--------|-----------------------|
| 100001 | 服务器内部错误，请重试           |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确 |

|        |               |
|--------|---------------|
| 100007 | 当前付费套餐包不支持该能力 |
|--------|---------------|



# 第三方回调

## 第三方回调简介

最近更新时间：2025-04-29 16:54:12

为了方便您控制 App 的功能形态，LiveKit 提供了回调能力。

### 功能说明

用户可通过 REST API 方式配置指定 Url 的回调，当执行的 CallbackCommand 在配置列表中，将触发回调。

### 注意事项

- 一个 sdkAppId 仅允许配置一个回调。
- 确保回调的 URL 是可以正常访问。

### 回调协议

第三方回调基于 HTTP/HTTPS 协议，App 后台需要向 live 平台 提供回调 URL，即时通信 IM 使用 POST 请求的方式来向 App 后台发起回调请求。即时通信 IM 在发起回调时，会在 App 提供的 URL 之后增加如下几个参数：

| 参数              | 含义                                                                                                                        |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|
| SdkAppid        | App 在即时通信 IM 分配的应用标识                                                                                                      |
| CallbackCommand | 回调命令字                                                                                                                     |
| contenttype     | 可选，通常值为 JSON                                                                                                              |
| ClientIP        | 客户端 IP 地址                                                                                                                 |
| OptPlatform     | 客户端平台，对应不同的平台类型，可能的取值有：RESTAPI（使用 REST API 发送请求）、Web（使用 Web SDK 发送请求）、Android、iOS、Windows、Mac、iPad、Unknown（使用未知类型的设备发送请求） |

### 安全问题

安全性问题：

1. HTTP 是明文传输，数据的保密性无法保证，建议使用 HTTPS。
2. 无法验证回调请求是否真正来自于 Live。

为了解决请求来源的安全性问题，我们为您提供回调应用层签名认证，启用步骤如下：

1. 首先使用 [设置回调配置](#) 设置回调。
2. 在 [设置回调鉴权 Token](#) 中，开启鉴权，并配置鉴权 Token，配置完成后，回调请求 URL 参数会增加签名 Sign 以及签名时间戳 RequestTime，签名算法：Sign=sha256(TokenRequestTime)。

3. 在 App 后台支持针对回调请求进行鉴权，通过本地鉴权 Token 以及回调 URL 参数中的签名时间戳 RequestTime 计算 sha256，比较签名是否匹配。并对 RequestTime 校验时效性，RequestTime 和当前时间相差超过1分钟即可认为是无效请求，以防止重放攻击。

#### 签名算法示例：

```
Token=xxxxxyyyy  
RequestTime=1669872112  
Sign=sha256(xxxxxyyy1669872112)=17773bc39a671d7b9aa835458704d2a6db81360a59  
40292b587d6d760d484061
```

#### 回调

```
URL=URL&Sign=17773bc39a671d7b9aa835458704d2a6db81360a5940292b587d6d760d484  
061&RequestTime=1669872112
```

## 回调不通的常见原因

如果遇到回调不通的情况，App 先依照如下清单排查一下设置的回调服务是否存在问题。

| 回调不通的现象           | 可能存在的原因                                                                                                                                                                                                                                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 回调 URL 访问超时       | <ol style="list-style-type: none"><li>1. Live无法完成 DNS 解析，请确认该域名是否在公网生效。（例如，回调 HOST 为 http://notexist.com，该域名不存在，Live无法完成 DNS 解析。）</li><li>2. Live无法访问到回调 URL 中配置的 IP，请确认该 IP 是否公网可达。（例如，回调 HOST 为 http://10.0.0.1，该域名为 App 内网 IP，Live无法访问到该 IP。）</li><li>3. App 回调服务防火墙策略限制，请检查防火墙配置。（例如，App 回调服务器拒绝了所有到达 80 端口的请求。）</li></ol> |
| 回调服务拒绝访问          | Live 可以访问到 HOST，但链接建立失败，请确认 WebServer 已经正确启动。（例如：App 回调服务器的 WebServer 并未启动，或者端口配置错误。）                                                                                                                                                                                                                                          |
| 回调服务 HTTP 返回码非200 | 回调请求成功，但应答报文中的 HTTP 返回码非200。                                                                                                                                                                                                                                                                                                   |
| 回调应答包体解析失败        | 回调请求包体非 JSON 格式。                                                                                                                                                                                                                                                                                                               |

# 回调命令列表

最近更新时间：2025-06-20 15:11:32

## 房间相关

| 回调类型          | 回调命令字                                                         |
|---------------|---------------------------------------------------------------|
| 创建房间之后回调      | <a href="#">Live.CallbackAfterCreateRoom</a>                  |
| 解散房间之后回调      | <a href="#">Live.CallbackAfterDestroyRoom</a>                 |
| 更新房间信息之后回调    | <a href="#">Live.CallbackAfterUpdateRoomInfo</a>              |
| 设置房间自定义信息之后回调 | <a href="#">Live.CallbackAfterSetMetadata</a>                 |
| 删除房间自定义信息之后回调 | <a href="#">Live.CallbackAfterDelMetadata</a>                 |
| 创建房间达到70%预警回调 | <a href="#">Live.CallbackAfterCreateRoomReachingThreshold</a> |

## 用户相关

| 回调类型       | 回调命令字                                                |
|------------|------------------------------------------------------|
| 房间成员状态变更回调 | <a href="#">Live.CallbackAfterMemberStateChanged</a> |

## 连麦相关

| 回调类型       | 回调命令字                                            |
|------------|--------------------------------------------------|
| 麦位列表变更之后回调 | <a href="#">Mic.CallbackAfterSeatInfoChanged</a> |

## PK 相关

| 回调类型       | 回调命令字                                          |
|------------|------------------------------------------------|
| 创建 PK 之后回调 | <a href="#">Live.CallbackAfterCreateBattle</a> |
| PK 开启之后回调  | <a href="#">Live.CallbackAfterStartBattle</a>  |
| PK 结束之后回调  | <a href="#">Live.CallbackAfterEndBattle</a>    |

# 回调配置

## 查询回调配置

最近更新时间：2025-04-03 17:08:43

### 功能说明

App 管理员可以通过该接口创建回调。

### 接口调用说明

#### 请求 URL 示例

```
https://xxxxxx/v4/live_config/get_callback?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

#### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                          | 说明                                                                                                                                                                              |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxxx                     | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>● 中国：console.tim.qq.com</li><li>● 新加坡：adminapisgp.im.qcloud.com</li><li>● 美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_config/get_callback | 请求接口                                                                                                                                                                            |
| sdkappid                    | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                    |
| identifier                  | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                   |
| usersig                     | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                               |
| random                      | 请输入随机的32位无符号整数，取值范围0 - 4294967295                                                                                                                                               |
| contenttype                 | 请求格式固定值为 json                                                                                                                                                                   |

#### 最高调用频率

200次/秒。

## 请求包示例

- 基础形式  
创建房间

```
{
}
```

## 请求包字段说明

## 应答包体示例

- 基础形式

```
{
  "ErrorCode": 0,
  "ErrorInfo": "",
  "ActionStatus": "OK",
  "RequestId": "Id-1cc8828fd3d84795ac866ced43b15b5c-O-Seq-61309",
  "Response": {
    "Url": "http://www.example.com/callback",
    "CallbackCommandList": [
      "Live.CallbackAfterCreateRoom",
      "Live.CallbackAfterDestroyRoom"
    ]
  }
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |
| Url          | String  | 回调的地址，需要以 http/https 开头，建议使用更为安全的 https   |

|                         |       |                                       |
|-------------------------|-------|---------------------------------------|
| CallbackComm<br>andList | Array | 需要触发回调的命令字，参见 <a href="#">回调命令字列表</a> |
|-------------------------|-------|---------------------------------------|

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                     |
|--------|--------------------------|
| 100001 | 服务器内部错误，请重试              |
| 100002 | 参数非法，请根据错误描述检查请求是否正确。    |
| 100301 | 回调配置不存在，可以时间创建回调配置接口进行创建 |

# 设置回调配置

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口设置回调配置。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_config/set_callback?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype  
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                          | 说明                                                                                                                                                                        |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| v4/live_config/set_callback | 请求接口                                                                                                                                                                      |
| xxxxxx                      | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国：console.tim.qq.com</li><li>新加坡：adminapisgp.im.qcloud.com</li><li>美国：adminapiusa.im.qcloud.com</li></ul> |
| sdkappid                    | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                              |
| identifier                  | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                             |
| usersig                     | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                         |
| random                      | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                         |
| contenttype                 | 请求格式固定值为 json                                                                                                                                                             |

### 最高调用频率

200次/秒。

### 请求包示例

- 基础形式

## 创建房间

```
{
  "Url": "http://www.example.com/callback",
  "CallbackCommandList": [
    "Live.CallbackAfterCreateRoom",
    "Live.CallbackAfterDestroyRoom",
    "Live.CallbackAfterMemberStateChanged"
  ]
}
```

## 请求包字段说明

| 字段                  | 类型     | 属性 | 说明                                      |
|---------------------|--------|----|-----------------------------------------|
| Url                 | String | 必填 | 回调的地址，需要以 http/https 开头，建议使用更为安全的 https |
| CallbackCommandList | Array  | 必填 | 需要触发回调的命令字，参见 <a href="#">回调命令字列表</a>   |

## 应答包体示例

- 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                          |
|--------------|---------|-----------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败            |



|           |        |                                           |
|-----------|--------|-------------------------------------------|
| ErrorInfo | String | 错误信息                                      |
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                     |
|--------|--------------------------|
| 100001 | 服务器内部错误，请重试              |
| 100002 | 参数非法，请根据错误描述检查请求是否正确。    |
| 100300 | 回调配置已存在，可以使用更新回调配置接口进行更新 |

# 更新回调配置

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口创建回调。

## 接口调用说明

### 请求 URL 示例

```
https://xxxxxx/v4/live_config/update_callback?  
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype  
=json
```

### 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                             | 说明                                                                                                                                                                        |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| xxxxxx                         | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国：console.tim.qq.com</li><li>新加坡：adminapisgp.im.qcloud.com</li><li>美国：adminapiusa.im.qcloud.com</li></ul> |
| v4/live_config/update_callback | 请求接口                                                                                                                                                                      |
| sdkappid                       | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                              |
| identifier                     | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                             |
| usersig                        | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                         |
| random                         | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                         |
| contenttype                    | 请求格式固定值为 json                                                                                                                                                             |

### 最高调用频率

200次/秒。

### 请求包示例

- 基础形式

## 创建房间

```
{
  "Url": "http://www.example.com/callback",
  "CallbackCommandList": [
    "Live.CallbackAfterCreateRoom",
    "Live.CallbackAfterDestroyRoom"
  ]
}
```

## 请求包字段说明

| 字段                  | 类型     | 属性 | 说明                                      |
|---------------------|--------|----|-----------------------------------------|
| Url                 | String | 必填 | 回调的地址，需要以 http/https 开头，建议使用更为安全的 https |
| CallbackCommandList | Array  | 必填 | 需要触发回调的命令字，参见 <a href="#">回调命令字列表</a>   |

## 应答包体示例

- 基础形式

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0,
  "RequestId": "Id-8c9858f01e954611ae2d4c1b1ed7d583-O-Seq-52720"
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                          |
|--------------|---------|-----------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败            |

|           |        |                                           |
|-----------|--------|-------------------------------------------|
| ErrorInfo | String | 错误信息                                      |
| RequestId | String | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                     |
|--------|--------------------------|
| 100001 | 服务器内部错误，请重试              |
| 100002 | 参数非法，请根据错误描述检查请求是否正确。    |
| 100301 | 回调配置不存在，可以时间创建回调配置接口进行创建 |

# 删除回调配置

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 管理员可以通过该接口删除回调配置。

## 接口调用说明

## 请求 URL 示例

```
https://xxxxxx/v4/live_config/delete_callback?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype
=json
```

## 请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参见 [REST API 简介](#)。

| 参数                             | 说明                                                                                                                                                                           |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| v4/live_config/delete_callback | 请求接口                                                                                                                                                                         |
| xxxxxx                         | SDKAppID 所在国家/地区对应的专属域名： <ul style="list-style-type: none"><li>中国： console.tim.qq.com</li><li>新加坡： adminapisgp.im.qcloud.com</li><li>美国： adminapiusa.im.qcloud.com</li></ul> |
| sdkappid                       | 创建应用时即时通信 IM 控制台分配的 SDKAppID                                                                                                                                                 |
| identifier                     | 必须为 App 管理员账号，更多详情请参见 <a href="#">App 管理员</a>                                                                                                                                |
| usersig                        | App 管理员账号生成的签名，具体操作请参见 <a href="#">生成 UserSig</a>                                                                                                                            |
| random                         | 请输入随机的32位无符号整数，取值范围0 – 4294967295                                                                                                                                            |
| contenttype                    | 请求格式固定值为 json                                                                                                                                                                |

## 最高调用频率

200次/秒。

## 请求包示例

- 基础形式

创建房间

```
{  
}
```

## 应答包体示例

- 基础形式

```
{  
  "ErrorCode": 0,  
  "ErrorInfo": "",  
  "ActionStatus": "OK",  
  "RequestId": "Id-1cc8828fd3d84795ac866ced43b15b5c-O-Seq-61309"  
}
```

## 应答包字段说明

| 字段           | 类型      | 说明                                        |
|--------------|---------|-------------------------------------------|
| ActionStatus | String  | 请求处理的结果，OK 表示处理成功，FAIL 表示失败               |
| ErrorCode    | Integer | 错误码，0表示成功，非0表示失败                          |
| ErrorInfo    | String  | 错误信息                                      |
| RequestId    | String  | 唯一请求 ID，每次请求都会返回，定位问题时需要提供该次请求的 RequestId |

## 错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。

公共错误码（60000到79999）参见 [错误码](#) 文档。

本 API 私有错误码如下：

| 错误码    | 含义说明                  |
|--------|-----------------------|
| 100001 | 服务器内部错误，请重试           |
| 100002 | 参数非法，请根据错误描述检查请求是否正确。 |

# 房间相关

## 创建房间之后回调

最近更新时间：2025-09-09 10:24:42

### 功能说明

App 后台可以通过该回调实时查看用户创建房间的信息，包括：App 后台有房间创建成功通知时，可以据此进行数据同步等操作。

### 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

### 可能触发该回调的场景

- App 用户通过客户端创建房间成功
- App 管理员通过 REST API 创建房间成功

### 回调发生时机

房间创建成功之后。

### 接口说明

#### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。  
示例：

```
https://www.example.com?  
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

#### 请求参数说明

| 参数            | 说明                     |
|---------------|------------------------|
| https         | 请求协议为 HTTPS，请求方式为 POST |
| www.example.c | 回调 URL                 |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| om              |                                                             |
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID                               |
| CallbackCommand | 固定为 Live.CallbackAfterCreateRoom                            |
| contenttype     | 固定值为 JSON                                                   |
| ClientIP        | 客户端 IP，格式如：127.0.0.1                                        |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterCreateRoom",
  "Operator_Account": "admin",
  "RoomInfo": {
    "RoomId": "tandy-test-rest",
    "RoomName": "tandy-test-rest",
    "RoomType": "Live",
    "Owner_Account": "user3",
    "MaxMemberCount": 300,
    "IsMessageDisabled": true,
    "CustomInfo": "custom123",
    "IsSeatEnabled": true,
    "TakeSeatMode": "ApplyToTake",
    "MaxSeatCount": 16,
    "CreateTime": 1703589922,
    "IsPublicVisible": true,
    "CoverURL": "cover url",
    "Category": [1, 2]
  },
  "EventTime": 1703589922780
}
```

请求包字段说明

| 字段              | 类型     | 说明   |
|-----------------|--------|------|
| CallbackCommand | String | 回调命令 |



|                   |         |                                                  |
|-------------------|---------|--------------------------------------------------|
| Operator_Account  | String  | 发起创建群组请求的操作者 UserID                              |
| RoomId            | String  | 房间 ID                                            |
| RoomName          | String  | 房间名称                                             |
| RoomType          | String  | 房间类型：Meeting（会议房间）                               |
| Owner_Account     | String  | 房主 ID                                            |
| MaxMemberCount    | Integer | 最大房间成员数量                                         |
| IsMessageDisabled | Boolean | 禁止所有成员发文本消息                                      |
| CustomInfo        | String  | 自定义字段                                            |
| IsSeatEnabled     | Boolean | 是否支持麦位                                           |
| MaxSeatCount      | Integer | 最大麦位数                                            |
| TakeSeatMode      | String  | 麦位模式：None（关闭），FreeToTake（自由上麦），ApplyToTake（申请上麦） |
| CreateTime        | Integer | 预定会议开始时间                                         |
| IsPublicVisible   | Boolean | 房间是否公开                                           |
| CoverURL          | String  | 房间封面 url                                         |
| Category          | Array   | 房间类型标识，整数数组类型                                    |
| EventTime         | Integer | 事件触发的毫秒级别时间戳                                     |

## 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |
| ErrorInfo    | String  | 必填 | 错误信息                        |

## 参考

- [第三方回调简介](#)
- REST API: [创建房间](#)

# 解散房间之后回调

最近更新时间：2025-04-29 16:54:12

## 功能说明

App 后台可以通过该回调实时查看房间的解散动态，包括：对房间的解散实时记录（例如记录日志，或者同步到其他系统）。

## 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

## 可能触发该回调的场景

- App 用户通过客户端解散房间成功。
- App 管理员通过 REST API 解散房间成功。

## 回调发生时机

房间销毁成功之后。

## 接口说明

### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。

示例：

```
https://www.example.com?  
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

### 请求参数说明

| 参数              | 说明                     |
|-----------------|------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST |
| www.example.com | 回调 URL                 |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID                               |
| CallbackCommand | 固定为 Live.CallbackAfterDestroyRoom                           |
| contenttype     | 固定值为 JSON                                                   |
| ClientIP        | 客户端 IP，格式如：127.0.0.1                                        |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterDestroyRoom",
  "Operator_Account": "admin",
  "RoomId": "tandy-test-rest",
  "EventType": "DestroyByUser", // "DestroyByUser", "DestroyBySystem" 分为
  用户主动解散和房间无人后系统自动解散两种
  "EventTime": 1703589922780,
  "RoomInfo": {
    "RoomName": "live name",
    "RoomType": "Live",
    "Owner_Account": 144115216631667826,
    "IsSeatEnabled": true,
    "TakeSeatMode": "FreeToTake",
    "MaxMemberCount": 400,
    "MaxSeatCount": 4,
    "Category": [1, 2, 3],
    "CustomInfo": "",
    "IsMessageDisabled": false,
    "CoverURL": "cover url",
    "ActivityStatus": 0,
    "IsPublicVisible": false,
    "ViewCount": 0,
    "BackgroundURL": "background url",
    "IsUnlimitedRoomEnabled": true
  }
}
```

请求包字段说明

| 字段 | 类型 | 说明 |
|----|----|----|
|----|----|----|

|                   |         |                                                      |
|-------------------|---------|------------------------------------------------------|
| CallbackCommand   | String  | 回调命令                                                 |
| Operator_Account  | String  | 发起销毁房间请求的操作者 UserID                                  |
| RoomId            | String  | 房间 ID                                                |
| EventType         | String  | 解散类型：分为主动解散(DestroyByUser)和系统自动解散(DestroyBySystem)两种 |
| EventTime         | Integer | 事件触发的毫秒级别时间戳                                         |
| RoomName          | String  | 房间名称                                                 |
| RoomType          | String  | 房间类型：Meeting（会议房间）                                   |
| Owner_Account     | String  | 房主 ID                                                |
| IsSeatEnabled     | Bool    | 是否支持麦位                                               |
| TakeSeatMode      | String  | 麦位模式：None（关闭），FreeToTake（自由上麦），ApplyToTake（申请上麦）     |
| MaxMemberCount    | Integer | 最大房间成员数量                                             |
| MaxSeatCount      | Integer | 最大麦位数                                                |
| Category          | Array   | 房间类型标识，整数数组类型                                        |
| CustomInfo        | String  | 自定义字段                                                |
| IsMessageDisabled | Bool    | 禁止所有成员发文本消息                                          |
| CoverURL          | String  | 房间封面 url                                             |

|                        |         |                                                                              |
|------------------------|---------|------------------------------------------------------------------------------|
| ActivityStatus         | Integer | 直播间活跃状态：用户自定义标记                                                              |
| IsPublicVisible        | Boolean | 房间是否公开                                                                       |
| ViewCount              | Integer | 用户进房总次数                                                                      |
| BackgroundURL          | String  | 房间背景图url                                                                     |
| IsUnlimitedRoomEnabled | Boolean | 是否开启房间混流，支持高并发场景，默认为 false。当为 true 时对应于 sdk 侧 <a href="#">TUILiveKit 视频房</a> |

### 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

### 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |
| ErrorInfo    | String  | 必填 | 错误信息                        |

### 参考

- [第三方回调简介](#)
- REST API: [解散房间](#)

# 更新房间信息之后回调

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 后台可以通过该回调实时查看房间信息的变更，包括对变动房间信息的实时记录（例如记录日志，或者同步到其他系统）。

## 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

## 可能触发该回调的场景

- App 用户通过客户端变动房间信息。
- App 管理员通过 REST API 变动房间信息。

## 回调发生时机

房间信息变动成功之后。

## 接口说明

### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。  
示例：

```
https://www.example.com?  
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

### 请求参数说明

| 参数              | 说明                     |
|-----------------|------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST |
| www.example.com | 回调 URL                 |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID                               |
| CallbackCommand | 固定为 <code>Live.CallbackAfterUpdateRoomInfo</code>           |
| contenttype     | 固定值为 JSON                                                   |
| ClientIP        | 客户端 IP，格式如：127.0.0.1                                        |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

## 请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterUpdateRoomInfo",
  "Operator_Account": "bob",
  "RoomInfo" : {
    "RoomId": "live-room",
    "RoomName" : "rname-123",
    "MaxMemberCount" : 300,
    "IsMessageDisabled" : false,
    "CoverURL": "https://xxxx.png",
    "Category": [1,2,3],
    "ActivityStatus":1,
    "IsPublicVisible": false,
    "TakeSeatMode": "ApplyToTake",
    "CustomInfo": "custom123"
  }
  "EventTime":1670574414123//毫秒级别，事件触发时间戳
}
```

## 请求包字段说明

| 字段               | 类型      | 说明                  |
|------------------|---------|---------------------|
| CallbackCommand  | String  | 回调命令                |
| Operator_Account | String  | 发起创建群组请求的操作者 UserID |
| RoomId           | String  | 房间 ID               |
| RoomName         | String  | 房间名称                |
| MaxMemberCount   | Integer | 最大房间成员数量            |



|                   |         |                                                  |
|-------------------|---------|--------------------------------------------------|
| IsMessageDisabled | Bool    | 禁止所有成员发文本消息                                      |
| CustomInfo        | String  | 自定义字段                                            |
| TakeSeatMode      | String  | 麦位模式：None（关闭），FreeToTake（自由上麦），ApplyToTake（申请上麦） |
| IsPublicVisible   | Bool    | 房间是否公开                                           |
| CoverURL          | String  | 房间封面 url                                         |
| Category          | Array   | 房间类型标识，整数数组类型                                    |
| ActivityStatus    | Integer | 直播间活跃状态：用户自定义标记                                  |
| EventTime         | Integer | 事件触发的毫秒级别时间戳                                     |

## 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |
| ErrorInfo    | String  | 必填 | 错误信息                        |

## 参考

- [第三方回调简介](#)
- REST API: [更新房间](#)

# 设置房间自定义信息后回调

最近更新时间：2025-06-20 15:11:32

## 功能说明

App 后台可以通过该回调实时查看房间 metadata 数据的设置。

## 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

## 可能触发该回调的场景

- App 用户通过客户端设置房间自定义信息。
- App 管理员通过 REST API 设置房间自定义信息。

## 回调发生时机

设置房间自定义信息成功之后。

## 接口说明

### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。  
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

### 请求参数说明

| 参数              | 说明                            |
|-----------------|-------------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST        |
| www.example.com | 回调 URL                        |
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| CallbackCommand | 固定为 <code>Live.CallbackAfterSetMetadata</code>              |
| contentType     | 固定值为 <code>json</code>                                      |
| ClientIP        | 客户端 IP，格式如： <code>127.0.0.1</code>                          |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

## 请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterSetMetadata",
  "Operator_Account": "brennanli",
  "RoomId": "live-room1111",
  "Metadata": [
    {
      "Key": "key1",
      "Value": "value1"
    },
    {
      "Key": "key2",
      "Value": "value2"
    }
  ],
  "EventTime": 1739965885831
}
```

## 请求包字段说明

| 字段               | 类型      | 说明                  |
|------------------|---------|---------------------|
| CallbackCommand  | String  | 回调命令                |
| Operator_Account | String  | 发起销毁房间请求的操作者 UserID |
| RoomId           | String  | 房间 ID               |
| Metadata         | Array   | metadata 数据         |
| Key              | String  | metadata 对应的 key    |
| Value            | String  | metadata 对应的 value  |
| EventTime        | Integer | 事件触发的毫秒级别时间戳        |

## 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorInfo    | String  | 必填 | 错误信息                        |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |

## 参考

- [第三方回调简介](#)
- [设置房间自定义信息](#)

# 删除房间自定义信息后回调

最近更新时间：2025-04-03 17:08:43

## 功能说明

App 后台可以通过该回调实时查看房间自定义信息的删除。

## 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

## 可能触发该回调的场景

- App 用户通过客户端删除 metadata。
- App 管理员通过 REST API 删除 metadata。

## 回调发生时机

房间 metadata 数据删除成功之后。

## 接口说明

### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。

示例：

```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

### 请求参数说明

| 参数              | 说明                            |
|-----------------|-------------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST        |
| www.example.com | 回调 URL                        |
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| CallbackCommand | Live.CallbackAfterDelMetadata                               |
| contentType     | 固定值为 json                                                   |
| ClientIP        | 客户端 IP，格式如：127.0.0.1                                        |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

## 请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterDelMetadata",
  "Operator_Account": "brennanli",
  "RoomId": "live-room1111",
  "Keys": [
    "key1", "key2"
  ],
  "EventTime": 1739966441121
}
```

## 请求包字段说明

| 字段               | 类型      | 说明                  |
|------------------|---------|---------------------|
| CallbackCommand  | String  | 回调命令                |
| Operator_Account | String  | 发起销毁房间请求的操作者 UserID |
| RoomId           | String  | 房间 ID               |
| Keys             | Array   | metadata 数据里对应的 Key |
| EventTime        | Integer | 事件触发的毫秒级别时间戳        |

## 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorInfo    | String  | 必填 | 错误信息                        |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |

## 参考

- [第三方回调简介](#)
- [删除房间自定义信息](#)

# 用户相关

## 房间成员状态变更回调

最近更新时间：2025-04-03 17:08:43

### 功能说明

App 后台可以通过该回调实时查看房间用户上下线状态变更。

### 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

### 可能触发该回调的场景

- App 用户通过客户端创建房间成功。
- App 管理员通过 REST API 创建房间成功。

### 回调发生时机

房间创建成功之后。

### 接口说明

#### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。  
示例：

```
https://www.example.com?  
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json
```

#### 请求参数说明

| 参数              | 说明                     |
|-----------------|------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST |
| www.example.com | 回调 URL                 |



|                 |                                          |
|-----------------|------------------------------------------|
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID            |
| CallbackCommand | 固定为 Live.CallbackAfterMemberStateChanged |
| contenttype     | 固定值为 JSON                                |

## 请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterMemberStateChanged",
  "RoomId": "room_id",
  "EventType": "Online", //Online, Offline 上下线两种
  "EventCause": "Enter", //分为以下四种 进房 Enter, 退房 Leave, 心跳断线
HeartbeatInterrupt, 心跳恢复 HeartbeatRecover
  "MemberList": [
    {
      "Member_Account": "jared"
    },
    {
      "Member_Account": "tommy"
    }
  ],
  "EventTime": 1703589922780
}
```

## 请求包字段说明

| 字段              | 类型     | 说明                                                                          |
|-----------------|--------|-----------------------------------------------------------------------------|
| CallbackCommand | String | 回调命令                                                                        |
| RoomId          | String | 房间 ID                                                                       |
| EventType       | String | 事件类型，分为房间成员上线和下线两种：Online，Offline                                           |
| EventCause      | String | 事件缘由，分为以下四种 进房 Enter，退房 Leave，心跳断线 HeartbeatInterrupt，心跳恢复 HeartbeatRecover |
| MemberList      | Array  | 涉及的成员列表                                                                     |
| Member_Account  | String | 账号 id                                                                       |

|           |         |              |
|-----------|---------|--------------|
| EventTime | Integer | 事件触发的毫秒级别时间戳 |
|-----------|---------|--------------|

## 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |
| ErrorInfo    | String  | 必填 | 错误信息                        |

## 参考

- [第三方回调简介](#)

# 连麦相关

## 麦位列表变更之后回调

最近更新时间：2025-05-20 17:56:32

### 功能说明

App 后台可以通过该回调实时查看麦位列表变更消息。

### 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

### 可能触发该回调的场景

- App 用户上下麦。
- 操作麦位锁定。

### 回调发生时机

麦位信息变更之后。

### 接口说明

#### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。

示例：

```
https://www.example.com?Sdkappid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

#### 请求参数说明

| 参数              | 说明                     |
|-----------------|------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST |
| www.example.com | 回调 URL                 |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID                               |
| CallbackCommand | 固定为 Mic.CallbackAfterSeatInfoChanged                        |
| contenttype     | 固定值为 JSON                                                   |
| ClientIP        | 客户端 IP，格式如：127.0.0.1                                        |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

## 请求包示例

```
{
  "CallbackCommand": "Mic.CallbackAfterSeatInfoChanged",
  "RoomId": "rid-123",
  "SeatList": [
    {
      // 麦位编号
      "Index": 1,
      // 如果当前麦位有人占用，则返回对应用户的 ID，否则该字段不会设置
      "Member_Account": "user2",
      // false:允许上麦 true:禁止上麦
      "IsTakenDisabled": false,
      // false:允许推视频流 true:禁止推视频流
      "IsVideoDisabled": false,
      // false:允许推音频流 true:禁止推音频流
      "IsAudioDisabled": false
    }
  ],
  "EventTime": 1670574414123 // 毫秒级别，事件触发时间戳
}
```

## 请求包字段说明

| 字段              | 类型      | 说明    |
|-----------------|---------|-------|
| CallbackCommand | String  | 回调命令  |
| RoomId          | String  | 房间 ID |
| SeatList        | Array   | 麦位列表  |
| Index           | Integer | 麦位号   |

|                 |         |                          |
|-----------------|---------|--------------------------|
| Member_Account  | String  | 麦位上的用户 ID，如果该麦没有用户，则没此字段 |
| IsTakenDisabled | Bool    | 麦位锁定                     |
| IsVideoDisabled | Bool    | 禁止麦位视频流                  |
| IsAudioDisabled | Bool    | 禁止麦位音频流                  |
| EventTime       | Integer | 事件触发的毫秒级别时间戳             |

## 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |
| ErrorInfo    | String  | 必填 | 错误信息                        |

## 参考

- [第三方回调简介](#)

# PK 相关

## 创建 PK 之后回调

最近更新时间：2025-04-03 17:08:44

### 功能说明

App 后台可以通过该回调实时查看用户创建 Battle 的信息。

### 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

### 可能触发该回调的场景

- App 用户通过客户端创建 battle 成功。
- App 管理员通过 REST API 创建 battle 成功。

### 回调发生时机

创建 battle 成功之后

### 接口说明

#### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。

示例：

```
https://www.example.com?  
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

#### 请求参数说明

| 参数              | 说明                     |
|-----------------|------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST |
| www.example.com | 回调 URL                 |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID                               |
| CallbackCommand | 固定为 <code>Live.CallbackAfterCreateBattle</code>             |
| contenttype     | 固定值为 <code>json</code>                                      |
| ClientIP        | 客户端 IP，格式如： <code>127.0.0.1</code>                          |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterCreateBattle",
  "Operator_Account": "brennanli",
  "BattleId": "4siHsNsWN/T3aub9zKraqPbSEGRX2z6gs3LDFi+d/3M=", //battle
id
  "FromRoomId": "pk-3", //发起battle的主叫方房间id
  "ToRoomIdList": [ //参与battle的被叫方房间id列表
    "pk-5"
  ],
  "Timeout": 30000, //如果NeedResponse为true，那么最多等待被叫方处理时长，如果
被叫方不处理表示放弃参加battle
  "Duration": 60000, //battle开启后的时长，单位毫秒
  "NeedResponse": false, //为false表示直接开始battle,如果为true得等被叫方处
理后才能开始battle
  "ExtensionInfo": "extension pk",
  "EventTime":1739954005000
}
```

请求包字段说明

| 字段               | 类型     | 说明                      |
|------------------|--------|-------------------------|
| CallbackCommand  | String | 回调命令                    |
| Operator_Account | String | 发起 battle 请求的操作者 UserID |
| BattleId         | String | PK 对应的 battle id        |
| FromRoomId       | String | battle 的主叫方房间 id        |
| ToRoomIdList     | Array  | 参与 battle 的被叫方房间 id 列表  |

|               |         |                                    |
|---------------|---------|------------------------------------|
| Timeout       | Integer | 创建 battle 后, 等待被叫方处理的最大等待时长, 单位 ms |
| Duration      | Integer | battle 的时长                         |
| NeedResponse  | Bool    | 创建 battle 后是否需要被叫方房主同意加入           |
| ExtensionInfo | String  | battle 的扩展信息                       |
| EventTime     | Integer | 事件触发的毫秒级别时间戳                       |

## 应答包示例

App 后台同步数据后, 发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型      | 属性 | 说明                            |
|--------------|---------|----|-------------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果, OK 表示处理成功, FAIL 表示失败 |
| ErrorCode    | Integer | 必填 | 错误码, 此处填0表示忽略应答结果             |
| ErrorInfo    | String  | 必填 | 错误信息                          |

## 参考

- [第三方回调简介](#)



# PK 开启之后回调

最近更新时间：2025-04-03 17:08:44

## 功能说明

App 后台可以通过该回调及时知晓 battle 开启。

## 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

## 可能触发该回调的场景

- battle 创建后，无需等待直接进入开启状态。
- battle 创建后，所有被叫方处理完 battle 请求后或者整体等待时间结束后，只要同意加入 battle 的被叫方房主大于等于1人，battle 正式开始。

## 回调发生时机

battle 正式开启后。

## 接口说明

### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。  
示例：

```
https://www.example.com?
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

### 请求参数说明

| 参数              | 说明                     |
|-----------------|------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST |
| www.example.com | 回调 URL                 |

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID                               |
| CallbackCommand | 固定为 <code>Live.CallbackAfterStartBattle</code>              |
| contenttype     | 固定值为 <code>json</code>                                      |
| ClientIP        | 客户端 IP，格式如： <code>127.0.0.1</code>                          |
| OptPlatform     | 客户端平台，取值参见 <a href="#">第三方回调简介：回调协议</a> 中 OptPlatform 的参数含义 |

请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterStartBattle",
  "BattleId": "4siHsNsWN/T3aub9zKraqI4zZAYPRpXQhdtKv1q4H0s=", // battle
id
  "Duration": 60000, // battle时长
  "CreateTime": 1739954005, // battle创建时间，单位秒
  "StartTime": 1739954005, // battle开启时间，单位秒
  "FromRoomInfo": { // battle 主叫方信息
    "RoomId": "pk-3",
    "Owner_Account": "brennan"
  },
  "ToRoomList": [ // battle 被叫方信息
    {
      "RoomId": "pk-5",
      "Owner_Account": "tandy"
    }
  ],
  "EventTime":1739954005000 //回调触发时间，单位毫秒
}
```

请求包字段说明

| 字段              | 类型      | 说明         |
|-----------------|---------|------------|
| CallbackCommand | String  | 回调命令       |
| BattleId        | String  | battle ID  |
| Duration        | Integer | battle 的时长 |

|               |         |                  |
|---------------|---------|------------------|
| CreateTime    | Integer | battle 创建时间      |
| StartTime     | Integer | battle 开启时间      |
| FromRoomInfo  | String  | battle 的主叫方信息    |
| ToRoomList    | Array   | 参与 battle 的被叫方信息 |
| RoomId        | String  | 房间 ID            |
| Owner_Account | String  | 房主账号 ID          |
| EventTime     | Integer | 事件触发的毫秒级别时间戳     |

### 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

### 应答包字段说明

| 字段           | 类型      | 属性 | 说明                          |
|--------------|---------|----|-----------------------------|
| ActionStatus | String  | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorInfo    | String  | 必填 | 错误信息                        |
| ErrorCode    | Integer | 必填 | 错误码，此处填0表示忽略应答结果            |

### 参考

- 第三方回调简介

# PK 结束之后回调

最近更新时间：2025-04-03 17:08:44

## 功能说明

App 后台可以通过该回调及时知晓 battle 结束。

## 注意事项

- 要启用回调，必须配置回调 URL，并打开本条回调协议对应的开关，配置方法详见 [第三方回调配置](#) 文档。
- 回调的方向是 Live 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

## 可能触发该回调的场景

- battle 正常到时间结束。
- battle 进行期间，所有房主退出 battle。
- battle 创建好后，因没有房主同意 battle 而无法进入开启模式直接结束。

## 回调发生时机

battle 结束后。

## 接口说明

### 请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`。

示例：

```
https://www.example.com?  
SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json&ClientIP=$ClientIP&OptPlatform=$OptPlatform
```

### 请求参数说明

| 参数              | 说明                     |
|-----------------|------------------------|
| https           | 请求协议为 HTTPS，请求方式为 POST |
| www.example.com | 回调 URL                 |

|                 |                                             |
|-----------------|---------------------------------------------|
| SdkAppid        | 创建应用时在即时通信 IM 控制台分配的 SDKAppID               |
| CallbackCommand | 固定为 Live.CallbackAfterEndBattle             |
| contenttype     | 固定值为 json                                   |
| ClientIP        | 客户端 IP，格式如： 127.0.0.1                       |
| OptPlatform     | 客户端平台，取值参见 第三方回调简介：回调协议 中 OptPlatform 的参数含义 |

请求包示例

```
{
  "CallbackCommand": "Live.CallbackAfterEndBattle",
  "BattleId": "4siHsNsWN/T3aub9zKraqPfGQAimPhdFoe6VWhtz9lY=",
  "Duration": 30000,
  "CreateTime": 1739956115,
  "StartTime": 1739956115,
  "EndTime": 1739956145,
  "OpType": 0, // 0表示时间到了正常结束，1表示battle开启后，所有主播都退出了导致的结束，2表示battle创建后，因无主播加入导致直接未开启就结束
  "FromRoomInfo": { //主叫方信息
    "RoomId": "pk-3",
    "Score": 0,
    "Owner_Account": 144115245353757792
  },
  "ToRoomList": [ //被叫方信息
    {
      "RoomId": "pk-5",
      "Score": 0,
      "Owner_Account": 144115245442327522
    }
  ],
  "EventTime": 1739956146119
}
```

请求包字段说明

| 字段              | 类型     | 说明   |
|-----------------|--------|------|
| CallbackCommand | String | 回调命令 |

|               |         |                                                                        |
|---------------|---------|------------------------------------------------------------------------|
| BattleId      | String  | battle ID                                                              |
| Duration      | Integer | battle 的时长                                                             |
| CreateTime    | Integer | battle 创建时间                                                            |
| StartTime     | Integer | battle 开启时间                                                            |
| EndTime       | Integer | battle 结束时间                                                            |
| OpType        | Integer | 0表示时间到了正常结束；1表示battle开启后，所有主播都退出了导致的结束；2表示 battle 创建后，因无主播加入导致直接未开启就结束 |
| FromRoomInfo  | String  | battle 的主叫方信息                                                          |
| ToRoomList    | Array   | 参与 battle 的被叫方信息                                                       |
| RoomId        | String  | 房间 ID                                                                  |
| Score         | Integer | 房间对应的 PK 分数                                                            |
| Owner_Account | String  | 房主 ID                                                                  |
| EventTime     | Integer | 事件触发的毫秒级别时间戳                                                           |

## 应答包示例

App 后台同步数据后，发送回调应答包。

```
{
  "ActionStatus": "OK",
  "ErrorInfo": "",
  "ErrorCode": 0 // 忽略回调结果
}
```

## 应答包字段说明

| 字段           | 类型     | 属性 | 说明                          |
|--------------|--------|----|-----------------------------|
| ActionStatus | String | 必填 | 请求处理的结果，OK 表示处理成功，FAIL 表示失败 |
| ErrorInfo    | String | 必填 | 错误信息                        |

|           |         |    |                  |
|-----------|---------|----|------------------|
| ErrorCode | Integer | 必填 | 错误码，此处填0表示忽略应答结果 |
|-----------|---------|----|------------------|

参考

- [第三方回调简介](#)

# 错误码（TUILiveKit）

最近更新时间：2025-04-08 14:37:02

## 客户端错误码

### 通用错误码

| 错误码   | 描述                                                      |
|-------|---------------------------------------------------------|
| 0     | 操作成功                                                    |
| -1    | 暂未归类的通用错误                                               |
| -2    | 请求被限频，请稍后重试                                             |
| -3    | 重复操作                                                    |
| -1000 | 未找到 SDKAppID，请在 <a href="#">腾讯云视立方 SDK 控制台</a> 确认应用信息   |
| -1001 | 调用 API 时，传入的参数不合法，检查入参是否合法                              |
| -1002 | 未登录,请调用 Login 接口                                        |
| -1003 | 获取权限失败，当前未授权音/视频权限，请查看是否开启设备权限                          |
| -1004 | 该功能需要开通额外的套餐，请在 <a href="#">腾讯云视立方 SDK 控制台</a> 按需开通对应套餐 |

### 本地用户渲染，视频管理，音频管理 API 回调错误定义

| 错误码   | 描述                       |
|-------|--------------------------|
| -1100 | 系统问题，打开摄像头失败。检查摄像头设备是否正常 |
| -1101 | 摄像头没有系统授权，检查系统授权         |
| -1102 | 摄像头被占用，检查是否有其他进程使用摄像头    |
| -1103 | 当前无摄像头设备，请插入摄像头设备解决该问题   |
| -1104 | 系统问题，打开麦克风失败。检查麦克风设备是否正常 |
| -1105 | 麦克风没有系统授权，检查系统授权         |
| -1106 | 麦克风被占用                   |
| -1107 | 当前无麦克风设备                 |



|       |                          |
|-------|--------------------------|
| -1108 | 获取屏幕分享对象失败，检查屏幕录制权限      |
| -1109 | 开启屏幕分享失败，检查房间内是否有人正在屏幕分享 |

## 房间管理相关 API 回调错误定义

| 错误码   | 描述                                                                         |
|-------|----------------------------------------------------------------------------|
| -2101 | 需要进房后才可使用此功能                                                               |
| -2102 | 房主不支持退房操作，Conference（会议）房间类型：可以先转让房主，再退房。<br>LivingRoom（直播）房间类型：房主只能解散房间   |
| -2103 | 当前房间类型下不支持该操作                                                              |
| -2105 | 创建房间 ID 非法，自定义 ID 必须为可打印 ASCII 字符（0x20-0x7e），最长48个字节                       |
| -2107 | 房间名称非法，名称最长30字节，字符编码必须是 UTF-8，如果包含中文                                       |
| -2108 | 当前用户已在别的房间内，单个 roomEngine 实例只支持用户进入一个房间，如果要进入不同的房间请先退房或者使用新的 roomEngine 实例 |

## 房间内用户信息 API 回调错误定义

| 错误码   | 描述     |
|-------|--------|
| -2200 | 未找到该用户 |

## 房间内用户发言管理 API 回调错误定义&房间内麦位管理 API 回调错误定义

| 错误码   | 描述                   |
|-------|----------------------|
| -2300 | 需要房主权限才能操作           |
| -2301 | 需要房主或者管理员权限才能操作      |
| -2310 | 信令请求无权限，例如取消非自己发起的邀请 |
| -2311 | 信令请求 ID 无效或已经被处理过    |
| -2312 | 信令请求重复               |
| -2340 | 最大麦位超出套餐包数量限制        |
| -2344 | 麦位编号不存在              |
| -2360 | 当前麦位音频被锁             |

|       |                                |
|-------|--------------------------------|
| -2361 | 需要向房主或管理员申请后打开麦克风              |
| -2370 | 当前麦位视频被锁, 需要由房主解锁麦位后, 才能打开摄像头  |
| -2371 | 需要向房主或管理员申请后打开摄像头              |
| -2372 | 当前麦位视频被锁, 需要由房主解锁麦位后, 才能打开屏幕分享 |
| -2373 | 需要向房主或管理员申请后打开屏幕分享             |
| -2380 | 当前房间已开启全员禁言                    |
| -2381 | 当前房间内, 您已被已禁言                  |

## 房间预加载API回调错误定义

| 错误码   | 描述         |
|-------|------------|
| -4001 | 当前房间不支持预加载 |

## REST API 公共错误码

| 错误码    | 含义说明                           |
|--------|--------------------------------|
| 6000 2 | HTTP 解析错误, 请检查 HTTP 请求 URL 格式  |
| 6000 3 | HTTP 请求 JSON 解析错误, 请检查 JSON 格式 |
| 6000 4 | 请求 URL 或 JSON 包体中账号或签名错误       |
| 6000 5 | 请求 URL 或 JSON 包体中账号或签名错误       |
| 6000 6 | SDKAppID 失效, 请核对 SDKAppID 有效性  |
| 6000 7 | REST 接口调用频率超过限制, 请降低请求频率       |
| 6000 8 | 服务请求超时或 HTTP 请求格式错误, 请检查并重试    |
| 6000 9 | 请求资源错误, 请检查请求 URL              |

|       |                                                                                                                                               |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 60010 | 请求需要 App 管理员权限                                                                                                                                |
| 60011 | SDKAppID 请求频率超限，请降低请求频率                                                                                                                       |
| 60012 | REST 接口需要带 SDKAppID，请检查请求 URL 中的 SDKAppID                                                                                                     |
| 60013 | HTTP 响应包 JSON 解析错误                                                                                                                            |
| 60014 | 置换账号超时                                                                                                                                        |
| 60015 | 请求包体账号类型错误，请确认账号为字符串格式                                                                                                                        |
| 60016 | SDKAppID 被禁用。                                                                                                                                 |
| 60017 | 请求被禁用。                                                                                                                                        |
| 60018 | 请求过于频繁，请稍后重试。                                                                                                                                 |
| 60019 | 请求过于频繁，请稍后重试。                                                                                                                                 |
| 60020 | 您的专业版套餐包已到期并停用，请登录 [即时通信 IM 购买页面] ( <a href="https://buy.cloud.tencent.com/avc">https://buy.cloud.tencent.com/avc</a> ) 重新购买套餐包。购买后，将在5分钟后生效。 |
| 60021 | RestAPI 调用来源 IP 非法。                                                                                                                           |

## 服务端错误码

### 说明：

- 以下错误码适用于2.0版本 SDK，公共错误码（60000到79999）参见 [错误码](#) 文档。
- 1.x版本 SDK 的服务端错误码请参见 [服务端错误码（旧）](#) 文档。

| 错误码    | 含义说明                     |
|--------|--------------------------|
| 100001 | 服务器内部错误，请重试              |
| 100002 | 请参数非法，请根据错误描述检查请求是否正确    |
| 100003 | 房间ID 已被使用，请选择别的房间ID      |
| 100004 | 房间不存在，或者曾经存在过，但是目前已经被解散。 |
| 100005 | 非房间成员                    |

|        |                                              |
|--------|----------------------------------------------|
| 100006 | 操作权限不足                                       |
| 100007 | 无付费信息，需在控制台购买套餐包                             |
| 100008 | 房间成员已满                                       |
| 100009 | 标签数量超上限                                      |
| 100010 | 房间 ID 已被使用，并且操作者为房主，可以直接使用                   |
| 100011 | 房间 ID 已被 IM 占用，可以换一个房间 ID 使用，或者先通过 IM 接口解散该群 |
| 100012 | 频率超过限制，例如创建房间超过频率超限，同一房间 ID，1秒内只能创建一次        |
| 100013 | 超过付费上限，例如麦位数，pk场次房间数量等超过付费限制                 |
| 100015 | 无效的房间类型                                      |
| 100016 | 该成员已经被封禁                                     |
| 100017 | 该成员已经被禁言                                     |
| 100018 | 当前房间需要密码才能进入                                 |
| 100019 | 进房密码错误                                       |
| 100020 | 管理员数量超过上限                                    |
| 100102 | 信令请求冲突                                       |
| 100200 | 麦位已锁定，可以尝试换一个麦位                              |
| 100201 | 当前麦位已经有人了                                    |
| 100202 | 已经处于排麦状态                                     |
| 100203 | 已经处于麦上状态                                     |
| 100204 | 没有在排麦列表中                                     |

|        |                         |
|--------|-------------------------|
| 100205 | 麦位已满                    |
| 100206 | 未在麦上                    |
| 100210 | 已经有用户在麦位上               |
| 100211 | 该房间不支持连麦                |
| 100251 | 连麦列表为空                  |
| 100400 | 当前连线不存在或结束              |
| 100401 | 该房间已经在连线中               |
| 100402 | 该房间存在待处理的连线请求           |
| 100403 | 当前房间与其他房间连线中            |
| 100404 | 超过连线和 battle 房间数量上限     |
| 100405 | 短时间内连线过于频繁，稍等一会再试       |
| 100411 | 该场次 battle 不存在或已结束      |
| 100412 | 发起的 battle 里没有一个有效的房间   |
| 100413 | 短时间内频繁发起 battle, 稍等一会再试 |
| 100414 | 该房间已经不在 battle 中        |
| 100415 | 该房间处于其他的 battle 场次中     |
| 100416 | 该房间存在待处理的 battle 请求     |
| 100419 | 该房间处于 battle 中          |
| 100420 | 该 battle 场次还未开始         |
| 100421 | 该 battle 场次已经结束         |
| 10050  | 房间 meta 数据中的 key 数量超过上限 |

|        |                                      |
|--------|--------------------------------------|
| 0      |                                      |
| 100501 | 房间 meta 数据中单个 key 对应的 val 超过最大字节数限制  |
| 100502 | 房间 meta数据中所有 key 对应的 val 总和超过最大字节数限制 |
| 100503 | 删除房间 meta 数据时候，被删除的 key 没有一个存在       |
| 100504 | 房间 meta 数据中的 key 大小超过了最大字节数限制        |

# 发布日志（TUILiveKit） iOS

最近更新时间：2025-04-03 17:08:44

## 2024年7月

| 发布动态          | 描述                                                                                                                                   | 发布时间       |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------|------------|
| Version 1.0.0 | <ul style="list-style-type: none"><li>支持自定义礼物</li><li>支持自定义弹幕</li><li>支持音效&amp;混响</li><li>支持点赞</li><li>支持麦控</li><li>支持房间列表</li></ul> | 2024.07.03 |

# Android

最近更新时间：2025-04-03 17:08:44

## 2024年7月

| 发布动态          | 描述                                                                                                                                   | 发布时间       |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------|------------|
| Version 1.0.0 | <ul style="list-style-type: none"><li>支持自定义礼物</li><li>支持自定义弹幕</li><li>支持音效&amp;混响</li><li>支持点赞</li><li>支持麦控</li><li>支持房间列表</li></ul> | 2024.07.03 |



# 常见问题（TUILiveKit）

## iOS

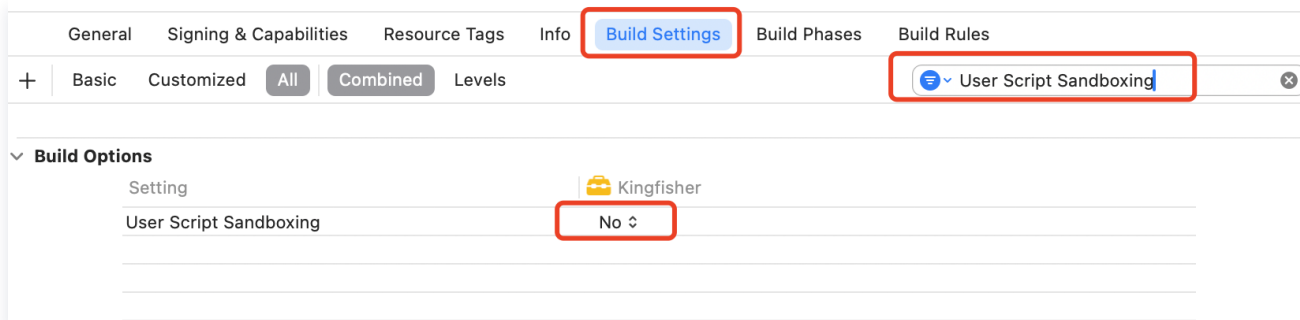
最近更新时间：2025-04-03 17:08:44

### Xcode 15 编译报错？

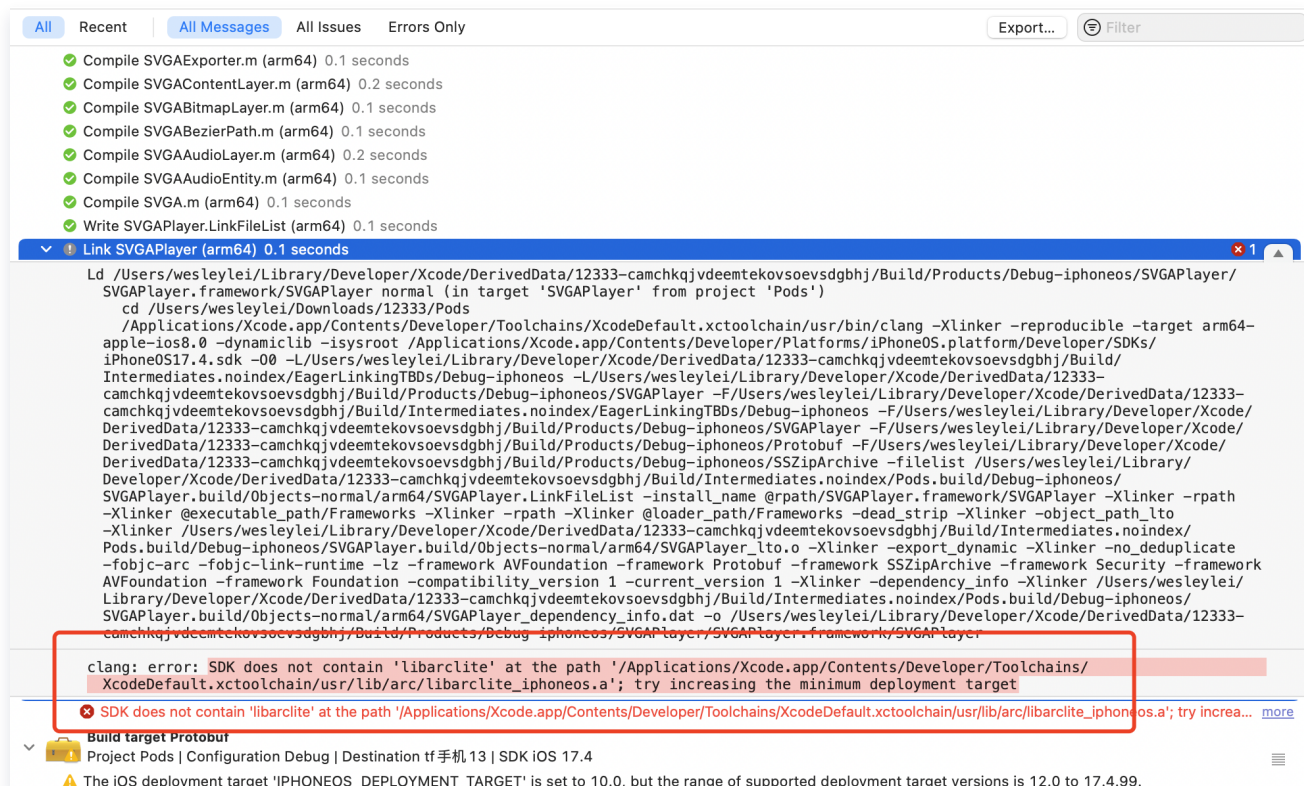
#### 1. 出现 Sandbox: rsync, 编译报错截图：



可以在 **Build Settings** 中把 **User Script Sandboxing** 设置为 **NO**：



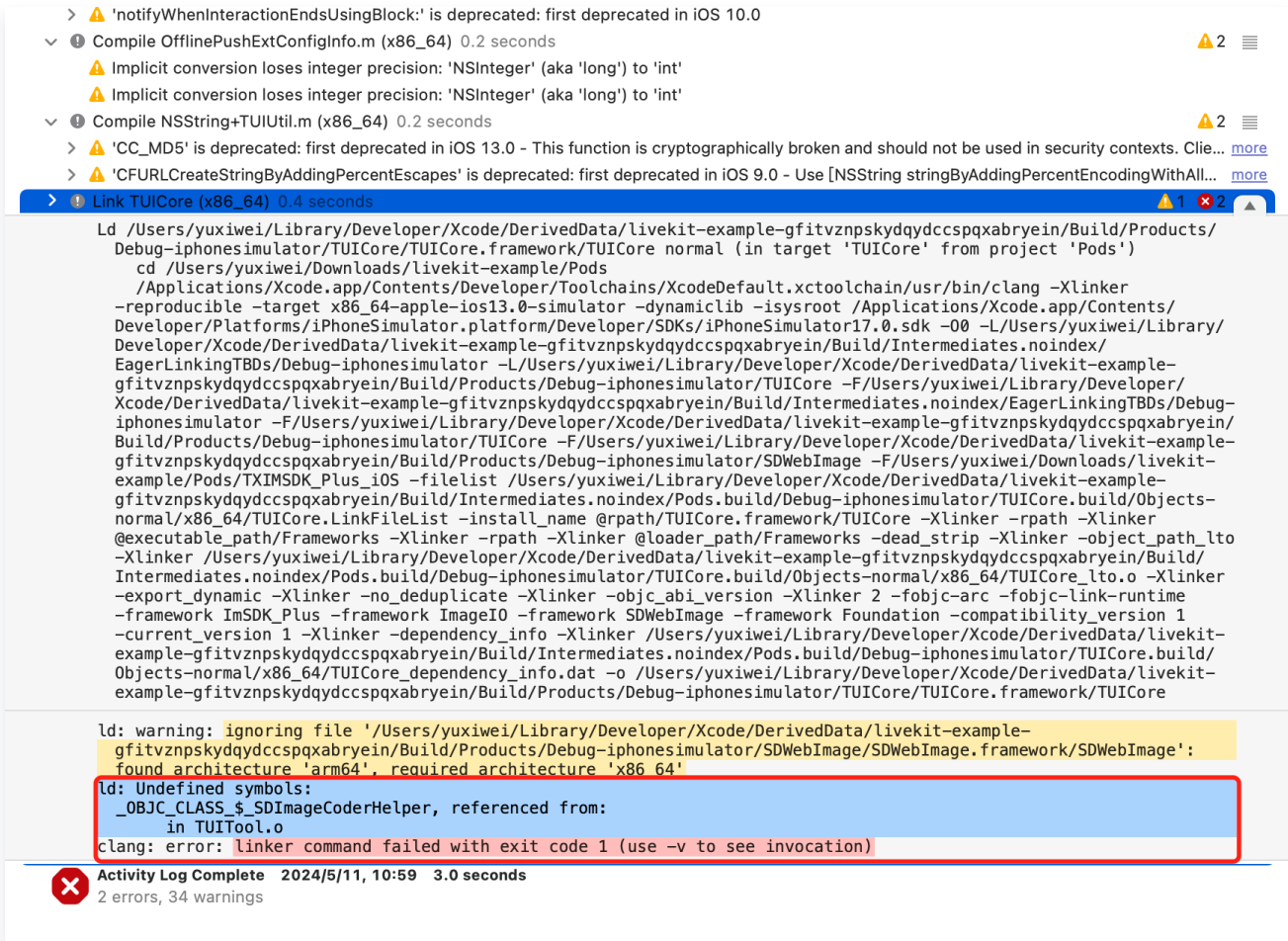
#### 2. 如果出现 SDK does not contain, 编译报错截图：



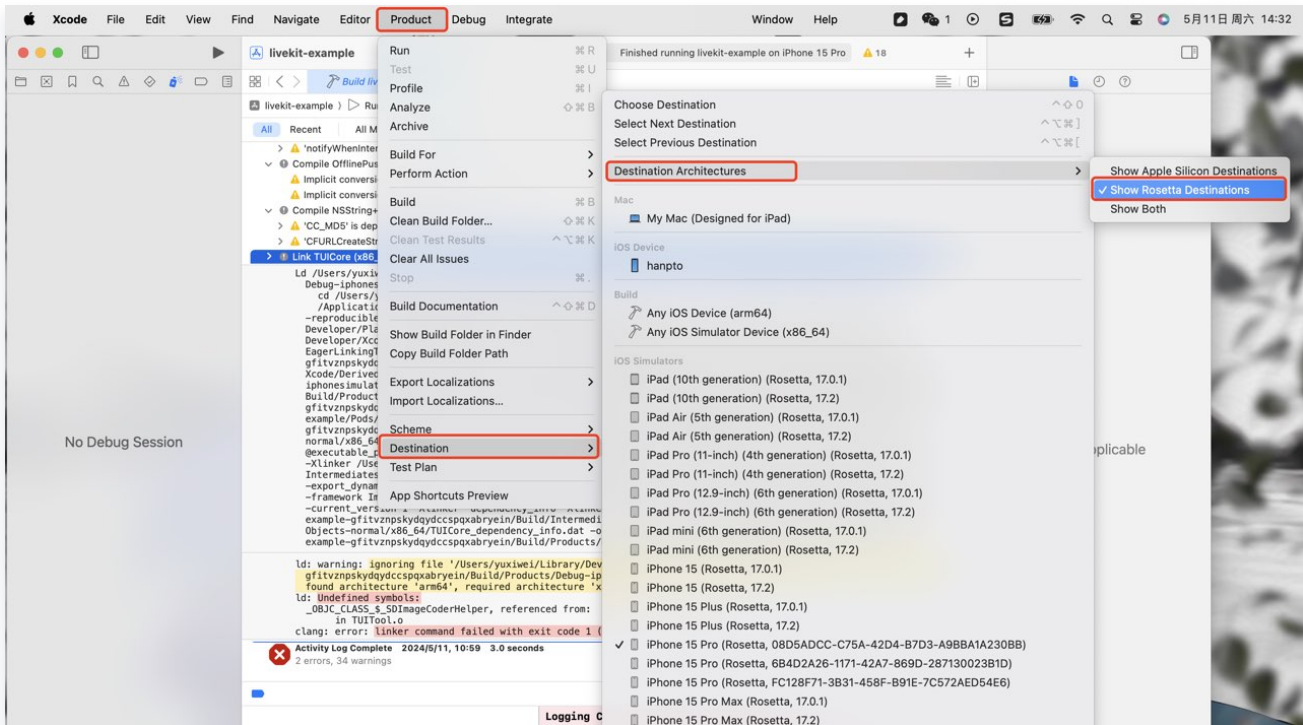
请在 Podfile 添加如下代码：

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    target.build_configurations.each do |config|
      config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '13.0'
    end
  end
end
```

3. 如果在 M 系列电脑上运行模拟器，可能会出现 Linker command failed with exit code 1 (use -v to see invocation)，编译报错截图：



需要修改 xcode 配置。xcode 打开项目 > Product > Destination > Destination Architectures 可以选择用哪种模式的模拟器打开，需要选择（Rosetta）结尾的模拟器。



## TUILiveKit 和自己集成的音视频库冲突了？

腾讯云的 音视频库 不能同时集成，可能存在符号冲突，可以按照下面的场景处理。

1. 如果您使用了 `TXLiteAVSDK_TRTC` 库，不会发生符号冲突。可直接在 `Podfile` 文件中添加依赖，

```
pod 'TUILiveKit'
```

2. 如果您使用了 `TXLiteAVSDK_Professional` 库，会产生符号冲突。您可在 `Podfile` 文件中添加依赖，

```
pod 'TUILiveKit/Professional'
```

3. 如果您使用了 `TXLiteAVSDK_Enterprise` 库，会产生符号冲突。建议升级到 `TXLiteAVSDK_Professional` 后使用 `TUILiveKit/Professional`。

## 如何查看 TRTC 日志？

TRTC 的日志默认压缩加密，后缀为 `.xlog`。日志是否加密是可以通过 `setLogCompressEnabled` 来控制，生成的文件名里面含 C(compressed) 的就是加密压缩的，含 R(raw) 的就是明文的。

iOS: `sandbox的Documents/log`

### ❗ 说明：

- 查看 `.xlog` 文件需要下载 [解密工具](#)，在 Python 2.7环境中放到 `xlog` 文件同目录下直接使用 `python decode_mars_log_file.py` 运行即可。
- 查看 `.clog` 文件（9.6 版本以后新的日志格式）需要下载 [解密工具](#)，在 Python 2.7 环境中放到 `clog` 文件同目录下直接使用 `python decompress_clog.py` 运行即可。

# Android

最近更新时间：2025-04-03 17:08:44

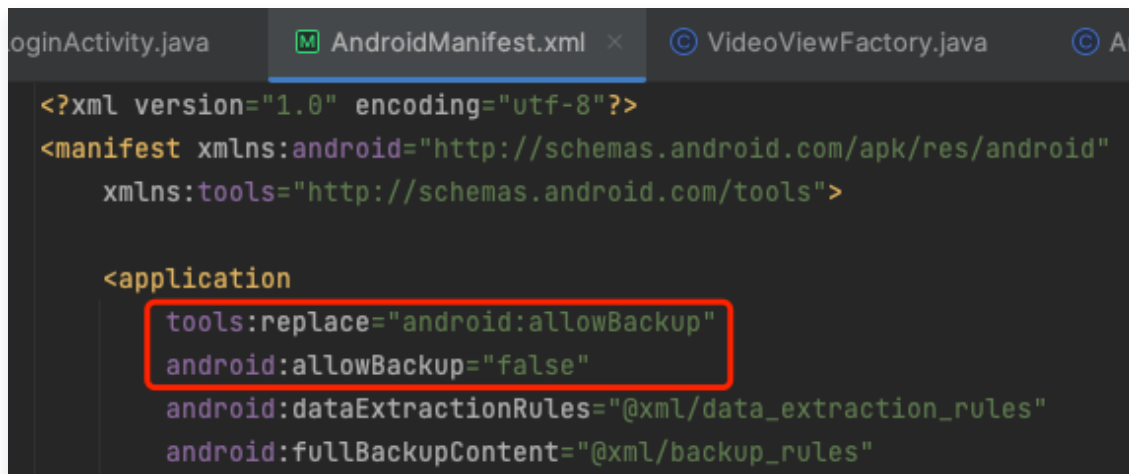
## TUILiveKit 是否可以不引入 IM SDK，只使用 TRTC？

不可以，TUIKit 全系组件都使用了腾讯云 IM SDK 作为通信的基础服务，例如创建房间信令、连麦信令等核心逻辑都使用 IM 服务，如果您已经购买有其他 IM 产品，也可以参照 TUILiveKit 逻辑进行适配。

## allowBackup 异常，如何处理？

```
Manifest merger failed : Attribute application@allowBackup value=(false) from AndroidManifest.xml:7:9-36
is also present at [com.github.yyued:SV6APlayer-Android:2.6.1] AndroidManifest.xml:12:9-35 value=(true).
Suggestion: add 'tools:replace="android:allowBackup"' to <application> element at AndroidManifest.xml:5:5-53:19 to override.
```

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



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        tools:replace="android:allowBackup"
        android:allowBackup="false"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
```

## Activity 主题问题：Activity need to use a Theme.AppCompat theme，如何处理？



```
FATAL EXCEPTION: main
Process: com.trtc.uikit.livekit.example, PID: 15190
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.trtc.uikit.livekit.example/com.trtc.uikit.livekit.example.login.LoginActivity}: java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) with this activity.
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3730)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3885)
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:101)
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2332)
    at android.os.Handler.dispatchMessage(Handler.java:107)
    at android.os.Looper.loop(Looper.java:230)
    at android.app.ActivityThread.main(ActivityThread.java:8115) <1 internal line>
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:526)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1034)
```

- **问题原因：**由于 LoginActivity 继承自 AppCompatActivity，所以要给 LoginActivity 设置一个 Theme.AppCompat 主题。
- **解决方法：**您可以在您工程的 AndroidManifest.xml 文件中 LoginActivity 的配置里，增加一个 Theme.AppCompat 主题。您也可以使用自己的 Theme.AppCompat 主题。修复示例如图所示：

```
<activity
    android:name=".login.LoginActivity"
    android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
</activity>
```

## 跳转浏览器打开网页地址失败？

解决办法：您可以在您工程的 AndroidManifest.xml 文件中，新增如下配置：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <queries>
        <intent>
            <action android:name="android.intent.action.VIEW" />
            <data android:scheme="https" />
        </intent>
    </queries>
```