

# 云联络中心 SDK 开发指南





【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】

## 🔗 腾讯云

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

SDK 开发指南 集成语音和在线工作台 SDK 使用 Demo 快速运行 Web uni-app Android iOS 初始化 SDK Web uni-app Android iOS 工作台 SDK API 文档 Web uni-app Android iOS 实现一键外呼 Web 小程序 uni-app Android iOS 实现电话呼入 Web 小程序/Android/iOS 常见问题 小程序端常见问题 Web SDK 常见问题 uni-app SDK 常见问题 客户端 SDK 常见问题 集成在线会话用户端 SDK Web (vue2/vue3) uni-app 微信小程序 Flutter iOS

腾讯云

Android

## SDK 开发指南 集成语音和在线工作台 SDK 使用 Demo 快速运行 Web

最近更新时间: 2023-10-11 11:39:32

我们提供了不同框架下的 Demo,您可以下载后快速运行:

- Vue Demo
- React Demo

下载完成后,根据 README.md 文档指引运行。您也可以继续根据后面的文档集成进您自己的项目。

## 交流与反馈

点此进入 TCCC 社群,享有专业工程师的支持,解决您的难题。



## uni-app

最近更新时间: 2025-06-30 19:46:11

本文主要介绍如何快速跑通腾讯云 TCCC uni-app Demo。

## 开发环境要求

- 建议使用最新的 HBuilderX 编辑器。
- iOS 9.0 或以上版本且支持音频的 iOS 设备。
- Android 版本不低于 4.1 且支持音频的 Android 设备,暂不支持模拟器。并请开启允许调试选项。
- iOS/Android 设备已经连接到 Internet。

## 前提条件

- 您已 注册腾讯云 账号,并完成 实名认证 。
- 您已 开通云联络中心 服务,并创建了 云联络中心实例 。
- 您已购买了号码,查看购买指南。并且完成了对应的 IVR 配置。

## 关键概念

- SdkAppld: 是用户在 腾讯云联络中心控制台 上创建的应用 ID,称之为 SdkAppld,一个腾讯云账号最多可以创建20个腾讯联络 中心应用,通常为140开头。
- UserID:座席或管理员在腾讯云联络中心内配置的账号,通常为邮箱格式,首次创建应用,主账号可前往站内信(子账号需订阅云联络中心产品消息)查看联络中心管理员账号和密码。一个 SDKAppID 下可以配置多个 UserID,如果超出配置数量限制,需到座席购买页购买更多座席数量。
- SecretId 和 SecretKey:开发者调用云 API 所需凭证,通过 腾讯云控制台 创建。
- Token: 登录票据,需要调用云 API 接口 CreateSDKLoginToken 来获取。正确的做法是将 Token 的计算代码和加密密钥放在 您的业务服务器上,然后由 App 按需向您的服务器获取实时算出的 Token。

## 操作步骤

#### 步骤1:下载 tccc-agent-uniapp-example 源码

根据实际业务需求 tccc-agent-uniapp-example 源码。

#### 步骤2:安装依赖

• 安装 npm 包依赖。

#### npm i tccc-sdk-uniapp

安装uni-ui。用 HBuilderX 导入uni-ui。



/ 前端组件 / 通用组件 / uni-ui		※ 1 选择一个uni-app项目导入该插件 播件版本: 14.27 の 場件版本: 14.27
uni-ui uni-ui uniui UHE194118 uHE198 uH	支持uni_modules	<sup>项目列表</sup> 第二步选择项目
uni-ui 是基于uni-app的、全端兼容的、高性能Ul框架	第一步	✓ tccc-agent-uniapp-example 未关疑uniCloud服务空间
作者: DCloud前端团队 V 🕕 《谢作者	冒 下载播件并导入HBuilderX	tccc-workstation-uni-demo 未关联uniCloud服务空间
下载人数: 155,835 下载次数: 877,802 🏠 收藏人数: 2,073	注意:每入uni_modules规范插件需要使用 3.1.0 以上版本的 HBuilderX	
***	■ 下载插件ZIP	第三步导入
插件ID: uni-ui		換定 取消 取消
插件包体积: 17.2KB	▲ 在线体验/部署	
更新日期: 2023-04-23 版本: 1.4.27		

## 步骤3:配置 tccc-agent-uniapp-example 工程文件

- 1. 找到并打开 debug/genTestToken.js 文件。
- 2. 设置 genTestToken.js 文件中的相关参数:
  - USERID: 座席账号,格式为: xxx@qq.com 。
  - SDKAPPID: 腾讯云联络中心 SDKAppId,需要替换为您自己账号下的 SDKAppId。
  - SECRETID: 计算签名用的加密密钥 ID。
  - SECRETKEY: 计算签名用的加密密钥 Key。

#### ▲ 注意:

请不要将如下代码发布到您的线上正式版本的 App 中,原因如下:



- 本文件中的代码虽然能够正确计算出 Token,但仅适合快速调通 SDK 的基本功能,不适合线上产品,这是因为客户端代码中的 SECRETKEY 很容易被反编译逆向破解,尤其是 Web 端的代码被破解的难度几乎为零。一旦您的密钥泄露,攻击者就可以计算出正确的 Token 来盗用您的腾讯云流量。
- 正确的做法是将 Token 的计算代码和加密密钥放在您的业务服务器上,然后由 App 按需向您的服务器获取实时算出的 Token。由于破解服务器的成本要高于破解客户端 App,所以服务器计算的方案能够更好地保护您的加密密钥。更多详情请 参见创建 SDK 登录 Token。

#### 步骤4:编译

使用**自定义基座打包运行**(不要选择标准基座运行),并且请使用**真机运行**自定义基座。

✓ 7HX5T19925011835	
<ul> <li>使用标准基座运行</li> <li>使用自定义基座运行</li> <li>什么是自定义基座</li> </ul>	
回名:com.tencent.tccc.uniplugin.demo 修改时间:2023/4/11 16:56:05 uniRu	intimeVersion:3.7.3 位置

#### ▲ 注意:

什么是自定义调试基座及使用说明,请参见 官方教程。

#### 步骤5:运行

- 1. 选择在**真机**运行后,单击**登录**。
- 2. 登录成功后输入需要拨打的手机号即可完成拨打功能。

#### 运行效果

基本功能如下图所示:

登录页面	号码管理页面	拨打页面
------	--------	------



子, 迎来到云呼叫中心 IID ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	075536564058 备注: 呼叫中心技术支持 录入时间: 2023.3.10	Ś	
Key			
IID 1400264214 🛞			
邮箱 gavinwjwang@tencent.com 📀			075536564058 <sub>通话中</sub>
			(1) 输入分机号
		Q	
登录	+ 添加外呼号码		

## 交流与反馈

点此进入 TCCC 社群,享有专业工程师的支持,解决您的难题。



## Android

最近更新时间: 2025-06-30 19:46:11

## 快速跑通腾讯云联络中心 Android Demo

腾讯云联络中心提供了 Android SDK,可以让座席通过固话话机进行通话。也可以通过我们提供的 SDK 来实现在手机端、PC 端外 呼、呼入来电接听等场景。

本文主要介绍如何快速跑通腾讯云联络中心 Android Demo,只要按照如下步骤进行配置,就可以跑通基于腾讯云联络中心相关功能。

## 开发环境要求

- Android Studio 3.5+。
- Android 4.1 (SDK API 16)及以上系统。

## 前提条件

- 您已 注册腾讯云 账号,并完成 实名认证 。
- 您已 开通云联络中心 服务,并创建了 云联络中心实例 。
- 您已购买了号码,查看购买指南。并且完成了对应的 IVR 配置

## 关键概念

- SdkAppld: 是用户在 腾讯云联络中心控制台 上创建的应用 ID,称之为 SdkAppld,一个腾讯云账号最多可以创建20个腾讯联络 中心应用,通常为140开头。
- UserID:座席或管理员在腾讯云联络中心内配置的账号,通常为邮箱格式,首次创建应用,主账号可前往站内信(子账号需订阅云联络中心产品消息)查看联络中心管理员账号和密码。一个 SDKAppID 下可以配置多个 UserID,如果超出配置数量限制,需到座席购买页购买更多座席数量。
- SecretId 和 SecretKey:开发者调用云 API 所需凭证,通过 腾讯云控制台 创建。
- Token: 登录票据,需要调用云 API 接口 CreateSDKLoginToken来获取。正确的做法是将 Token 的计算代码和加密密钥放在 您的业务服务器上,然后由 App 按需向您的服务器获取实时算出的 Token。

## 操作步骤

#### 步骤1: 下载 tccc-agent-java-example 源码

根据实际业务需求 tccc-agent-java-example 源码。

## 步骤2: 配置 tccc-agent-java-example 工程文件

- 1. 找到并打开 debug/src/main/java/com/tencent/tcccsdk/debug/GenerateTestUserToken.java 文件。
- 2. 设置 GenerateTestUserToken.java 文件中的相关参数:
  - USERID: 座席账号,格式为: xxx@qq.com。
  - SDKAPPID: 腾讯云联络中心 SDKAppId,需要替换为您自己账号下的 SDKAppId。
  - SECRETID: 计算签名用的加密密钥 ID。
  - SECRETKEY: 计算签名用的加密密钥 Key。



tccc-agent-java-example [TCCC Agent De	19	public class GenerateTestUserToken {
> 🖿 .gradle	28	
> 🛅 .idea	21	- /**
> 🔤 app	22	* 应度能是 终式为 : YYYGgg com
🗸 📷 debug	22	* 12.17XL 3, 10.477 * AAAByq.com
> 🖿 build	23	authlic static final Stains WSEDID - ""
libs	24	public static final String USERID = "";
Y 🖿 src	25	
> int androidTest	26	- /**
🗠 🌇 main	27	* 腾讯云呼叫中心 SDKAppId, 需要替换为您自己账号下的 SDKAppId。
🗸 🖿 java	28	* 进入腾讯云呼叫中心[控制台](https://console.cloud.tencent.com
Com.tencent.tcccsdk.debug	29	* 它是腾讯云用于区分客户的唯一标识。
GenerateTestUserToken	30	⊖ */
AndroidManifest.xml	31	public static final long <i>SDKAPPID</i> = 0;
> lest [unitTest]	32	
ild.gitignore	33	. /**
// build.gradle	34	* 计算签名用的加密密钥ID. [查看秘钥](https://console.cloud.tenc
consumer-rules.pro	35	* 注音:该方案仅沃田干调试Damo 正式上线前请您 UserSin 计管代码和率
proguard-rules.pro	34	+ 立地· https://cloud tapcant com/document/opoduct/670/5026
> gradie	30	* XH- nccps.//ccou.cencenc.com/uocomenc/producc/0///3626
.gitignore	57	
no api.mo	38	public static final String SECRETID = "";
Julid.gradle	39	
gradie.properties	40	0 /**
aradlew bat	41	* 计算签名用的加密密钥Key, [查看秘钥](https://console.cloud.ten
	42	* 注意: 该方案仅适用于调试Demo, 正式上线前请将 UserSig 计算代码和密
	43	* 文档: https://cloud.tencent.com/document/product/679/5826
A QuickStartDemo.md	44	⊖ */
A QuickStartSDK md	45	<pre>public static final String SECRETKEY = "";</pre>
an workstartspr.mu		

#### △ 注意:

请不要将如下代码发布到您的线上正式版本的 App 中,原因如下:

- 本文件中的代码虽然能够正确计算出 Token,但仅适合快速调通 SDK 的基本功能,不适合线上产品,这是因为客户端代 码中的 SECRETKEY 很容易被反编译逆向破解,尤其是 Web 端的代码被破解的难度几乎为零。一旦您的密钥泄露,攻 击者就可以计算出正确的 Token 来盗用您的腾讯云流量。
- 正确的做法是将 Token 的计算代码和加密密钥放在您的业务服务器上,然后由 App 按需向您的服务器获取实时算出的 Token。由于破解服务器的成本要高于破解客户端 App,所以服务器计算的方案能够更好地保护您的加密密钥。更多详 情请参见创建 SDK 登录 Token。

#### 步骤3:编译运行

使用 Android Studio (3.5及以上的版本) 打开源码工程 tccc-agent-java-example , 单击运行即可。

- 1. 单击**登录。**
- 2. 登录成功后输入需要拨打的手机号即可完成拨打功能。

#### 运行效果

基本功能如下图所示:

呼叫效果

接听效果





## 交流与反馈

点此进入 TCCC 社群,享有专业工程师的支持,解决您的难题。



## iOS

最近更新时间: 2025-06-30 19:46:11

## 快速跑通腾讯云联络中心 iOS Demo

腾讯云联络中心提供了 iOS SDK,可以让座席实现拨打电话、手机等功能。也可以通过我们提供的 SDK 来实现在手机端、PC 端外 呼、呼入来电接听等场景。

本文主要介绍如何快速跑通腾讯云联络中心 iOS Demo,只要按照如下步骤进行配置,就可以跑通基于腾讯云联络中心相关功能。

## 开发环境要求

- Xcode 9.0+。
- iOS 9.0 以上的 iPhone 或者 iPad 真机。
- 项目已配置有效的开发者签名。

## 前提条件

- 您已 注册腾讯云 账号,并完成 实名认证 。
- 您已 开通云联络中心 服务,并创建了 云联络中心实例 。
- 您已购买了号码,查看购买指南。并且完成了对应的 IVR 配置

## 关键概念

- 1. SdkAppld: 是用户在 腾讯云联络中心控制台 上创建的应用 ID,称之为 SdkAppld,一个腾讯云账号最多可以创建20个腾讯联络 中心应用,通常为140开头。
- 2. UserID: 座席或管理员在腾讯云联络中心内配置的账号,通常为邮箱格式,首次创建应用,主账号可前往站内信(子账号需订阅云联 络中心产品消息)查看联络中心管理员账号和密码。一个 SDKAppID 下可以配置多个 UserID,如果超出配置数量限制,需到座席 购买页购买更多座席数量。
- 3. SecretId 和 SecretKey:开发者调用云 API 所需凭证,通过 腾讯云控制台 创建。
- 4. Token: 登录票据,需要调用云API接口 CreateSDKLoginToken 来获取。正确的做法是将 Token 的计算代码和加密密钥放在 您的业务服务器上,然后由 App 按需向您的服务器获取实时算出的 Token。

#### 操作步骤

### 步骤1: 下载 tccc-agent-ios-example 源码

根据实际业务需求下载 tccc-agent-ios-example 源码。

#### 步骤2: 配置 tccc-agent-ios-example 工程文件

- 1. 找到并打开 debug/GenerateTestUserToken.h 文件。
- 2. 设置 GenerateTestUserToken.h 文件中的相关参数:
  - USERID: 座席账号,格式为: xxx@qq.com
  - SDKAPPID: 腾讯云联络中心 SDKAppId,需要替换为您自己账号下的 SDKAppId
  - SECRETID: 计算签名用的加密密钥 ID。
  - SECRETKEY: 计算签名用的加密密钥 Key。





#### △ 警告:

请不要将如下代码发布到您的线上正式版本的 App 中,原因如下:

- 本文件中的代码虽然能够正确计算出 Token,但仅适合快速调通 SDK 的基本功能,不适合线上产品,这是因为客户端代码 中的 SECRETKEY 很容易被反编译逆向破解,尤其是 Web 端的代码被破解的难度几乎为零。一旦您的密钥泄露,攻击者 就可以计算出正确的 Token 来盗用您的腾讯云流量。
- 正确的做法是将 Token 的计算代码和加密密钥放在您的业务服务器上,然后由 App 按需向您的服务器获取实时算出的 Token。由于破解服务器的成本要高于破解客户端 App,所以服务器计算的方案能够更好地保护您的加密密钥。更多详情请 参见创建 SDK 登录 Token

#### 步骤3:编译运行

使用 Xcode 打开源码工程 tccc-agent-ios-example , 单击运行即可。

- 1. 单击**获取 token > 登录**,
- 2. 登录成功后单击 外呼 即可完成拨打功能。

## 运行效果

基本功能如下图所示



11:14
请输入坐席ID
请输入先获取token
获取token 登录 isLogin 退出
外呼 挂断
静音 扬声器
1.5.1020

## 交流与反馈

点此进入 TCCC 社群,享有专业工程师的支持,解决您的难题。



## 初始化 SDK Web

最近更新时间: 2025-06-10 16:58:21

## 原理

云联络中心提供 JavaScript SDK 给开发者,开发者将 SDK 以 script 方式引入到页面中,即完成 SDK 的初始化,集成交互图如 下:



## 注意事项

- 1. TCCC 座席端 Web SDK 主要支持 Chrome 56版本及以上、Edge 80版本及以上的浏览器,建议安装最新版本的浏览器以支持更 多功能。
- 2. 请使用 HTTPS 协议来部署前端页面(开发时可以用 localhost ),否则会因为浏览器限制无法正常通话。

## 接入前提

- 1. 已创建云联络中心应用
- 2. 购买并添加 座席账号

## 关键概念

- SdkAppld:是腾讯云用户在 腾讯云联络中心控制台 上创建的应用 ID,称之为 SdkAppld,一个腾讯云账号最多可以创建50个腾 讯联络中心应用。
- UserID:座席或管理员在腾讯云联络中心内配置的账号,通常为邮箱格式,首次创建应用,主账号可前往站内信(子账号需订阅云 联络中心产品消息)查看联络中心管理员账号和密码。一个 SDKAppID 下可以配置多个 UserID,如果超出配置数量限制,需到座 席购买页购买更多座席数量。
- SecretId 和 SecretKey:开发者调用云 API 所需凭证,通过 腾讯云控制台 创建。



- SDKURL: 初始化 Web SDK 时的 JS URL,通过云 API 创建,该 URL 有效时长为10分钟,请确保只使用一次,在需要初始化 SDK 时请求创建,SDK 初始化成功后无需重复创建。
- SessionId:用户从开始接入到结束过程中的唯一ID,通过 SessionId,开发者可以关联不同的录音、服务记录和事件推送等。

## 步骤1:获取必备参数

- 1. 获取腾讯云账号的 SecretId 和 SecretKey ,您可参见 获取密钥。
- 2. 获取云联络中心应用的 sdkAppld 登录 云联络中心控制台 即可查看:

🔗 腾讯云 🛛 🕫	□ 范/*品 ×	提素产品、文档
云联络中心	电话呼入/呼出 如您在使用过程中递到任何问题,欢迎随时联系我们,咨询热线:0755-38564058,或点击加入TCCC技术服务交流群。	C 产品能力体验 C 联络中心 E
☞ 产品体验	+ 新地应用	
田 应用中心 ^ 电话呼入/呼出	应用記畫进展	应用名称 sdk4ppid 1400827822 行
<ul> <li>         ・ 在线客服         ・         ・         ・</li></ul>	申请等初接入 自有等码接入	
□ 企业资质管理	购买座席和通信套餐包 的运 请先购买通信套餐包用于抵扣固括树出400所入产生的费用:购买座席用于获取座席登录资格。	
」 座席官理 号码管理 日 奇祭句第理	<b>生效連席数量0个</b> 购买的還常数量为可支持的最大上线道席数量(该数量为管理员和普通使席数量之和)。	
<ul> <li>日 要素 2 日本</li> <li>記 数据与录音管理</li> <li>命 语音识别答理</li> </ul>	购买座席数量	
	制余面话客餐包分钟数 0 分钟, 特中调面话等码数 0 个 购买面话外评者管包后,方可进行电话评出(面话研入无需能外付费)。 购买面话呼出套餐包 购买面话号码	

## 步骤2: 获取 SDK URL

① 说明: 该步骤需要接入方后台开发实现。	
-----------------------	--

1. 引入 tencentcloud-sdk,引入方式可参见 SDK 中心 对应语言。

- 2. 调用接口 CreateSDKLoginToken。
- 3. 将获取到的 SdkURL 返回给接入方前端。

下文将使用接口名称 /loginTCCC 来说明该步骤开发的接口。以下代码以 Node.js 为例,其他语言示例代码请参见 CreateSDKLoginToken 。

```
// tencentcloud-sdk-nodejs的版本要求大于或等于4.0.3
const tencentcloud = require('tencentcloud-sdk-nodejs');
const express = require('express');
const app = express();
const CccClient = tencentcloud.ccc.v20200210.Client;
app.use('/loginTCCC', (req, res) => {
    const clientConfig = {
    // secret获取地址: https://console.cloud.tencent.com/cam/capi
    credential: {
        secretId: 'SecretId',
        secretKey: 'SecretKey'
    },
```



```
region: '',
profile: {
    httpProfile: {
        endpoint: 'ccc.tencentcloudapi.com'
    }
    };
    const client = new CccClient(clientConfig);
    const params = {
        SdkAppId: 140000000, // 请替换为自己的 SdkAppId
        SeatUserId: 'xxx@qq.com' // 替换为座席账号
    };
    client.CreateSDKLoginToken(params).then(
        (data) => {
            res.send({
                SdkURL: data.SdkURL
            })
        },
        (err) => {
            console.error('error', err);
            res.status(500);
        }
     );
})
```

## 步骤3:在 Web 前端请求获取 SDK URL 并完成初始化

说明:该步骤需要接入方前端开发进行。

- 1. 请求第二步实现的 /loginTCCC 接口,得到 SdkURL。
- 2. 将 SdkURL 以 script 方式插入页面。
- 3. 监听事件 tccc.events.ready 成功后,执行业务逻辑。



```
* Toco SDK 初始化成功,此时可调用外呼、监听呼入事件等功能。
 * 注意:请确保只初始化一次 SDK
 * */
 resolve('初始化成功')
 });
 window.tccc.on(window.tccc.events.tokenExpired, ({message}) => {
    console.error('初始化失败', message)
    reject(message)
    })
  })
  })
}
// 请求第二步突现的接口 /loginTCCC
// 注意: 以下仅为代码样例,不建议直接运行
fetch('/loginTCCC')
 .then(res => res.json())
 .then(res => f
    const SdkURL = res.SdkURL; // 请确保 SdkURL 都是通过请求返回的,否则可能会出现不可预知的错
[g]
    return injectTccCWebSdk(SdkURL);
  })
  .catch((error) => {
    // 初始化失败
    console.error(error);
  })
```



## uni-app

最近更新时间: 2025-06-30 19:46:11

本文主要介绍如何快速地将腾讯云 TCCC uni-app SDK 集成到您的项目中。

## 开发环境要求

- 建议使用最新的 HBuilderX 编辑器。
- iOS 9.0 或以上版本且支持音频的 iOS 设备。
- Android 版本不低于 4.1 且支持音频的 Android 设备,暂不支持模拟器。并请开启允许调试选项。
- iOS/Android 设备已经连接到 Internet。

## 接入前提

- 您已 注册腾讯云 账号,并完成 实名认证 。
- 您已 开通云联络中心 服务,并创建了 云联络中心实例 。
- 您已购买了号码,查看购买指南。并且完成了对应的 IVR 配置。

## 关键概念

- 1. SdkAppld: 是用户在 腾讯云联络中心控制台 上创建的应用 ID,称之为 SdkAppld,一个腾讯云账号最多可以创建20个腾讯联络 中心应用,通常为140开头。
- 2. UserID: 座席或管理员在腾讯云联络中心内配置的账号,通常为邮箱格式,首次创建应用,主账号可前往站内信(子账号需订阅云 联络中心产品消息)查看联络中心管理员账号和密码。一个 SDKAppID 下可以配置多个 UserID,如果超出配置数量限制,需到座 席购买页 购买更多座席数量。
- 3. SecretId 和 SecretKey:开发者调用云 API 所需凭证,通过 腾讯云控制台 创建。
- 4. token: 登录票据,需要调用云API接口 CreateSDKLoginToken 来获取。正确的做法是将 Token 的计算代码和加密密钥放在您 的业务服务器上,然后由 App 按需向您的服务器获取实时算出的 Token。

## 集成 SDK

1. 通过 npm 方式将 TCCC SDK 集成到您 uni-app 项目中。

npm i tccc-sdk-uniapp

 购买 uni-app SDK 插件:登录 uni 原生插件市场,在插件详情页中购买(免费插件也可以在插件市场0元购)。购买后才能够云端 打包使用插件。购买插件时请选择正确的 appid,以及绑定正确包名。





- 3. 配置权限:编辑 manifest.json 文件,配置麦克风权限,具体如下:
- iOS 需要以下权限: Privacy Microphone Usage Description,并填入麦克风使用目的提示语。

App启动界面配置	麦克风(NSMicrophoneUsageDescription)
App模块配置	需要访问你的麦克风权限,开启后才可正常通话。
	运行期访问位置(NSLocationWhenInUseUsageDescription)
App权限配直	
App原生插件配置	

Android 需要以下权限:

4. 配置音频后台运行:手机应用程序在切换到后台时,操作系统会暂停应用程序的进程以节省资源。这意味着应用程序的所有活动都将被 停止,包括播放音频。而ios下需要配置 audio background mode 才可以保证有音频影响的时候程序不会终止。



manifest ison	
基础配置	支持CPU类型
App图标配置	参考文档 ▼ armeabi-v7a
App启动界面配置	✓ arm64-v8a
· ·*···	<u> </u>
App惧状配直	i0S设置
App权限配置	UrlSchemes
App百生场供配署	注册schema在其它App中打开当前App,多个scheme使用','号分割,例如:test1,test;
App常用其它设置	关联域 (Associated Domains)
Web配置	用于配置通用链接域名,通用链接域名格式为: applinks:域名,例如: applinks:demo.(
微信小程序配置	
百度小程序配置	应用访问白名单
ᄚᇼᆊᇞᆂᆡᇧᇊᆇᆓᆿᄪ	允许当前App访问(查询是否安装、直接打开)的其它App白名单列表,填写其它App注册的s
字节跳动小桯序配置	
支付宝小程序配置	
QQ小程序配置	应用后台猫放自示(aud 10)、走位(tocation)等功能,多项使用",亏分割,例如:aud

### ▲ 注意:

不配置该权限,通话中切后台的时候会自动中断。

5. 使用**自定义基座打包运行**(不要选择标准基座运行),并且请使用**真机运行**自定义基座。

运行项目 [tccc-workstation-uni-demo] 到 Android 设备	^
€ 刷新	
✓ 7HX5T19925011835	
<ul> <li>○使用标准基座运行</li> <li>◇使用自定义基座运行</li> <li>什么是自定义基座</li> </ul>	
包名: com.tencent.tccc.uniplugin.demo 修改时间: 2023/4/11 16:56:05 uniRuntimeVersion:3.7.3 位	[置
故障排查指南 运行 运行	
▲ 注意:	



#### 什么是自定义调试基座及使用说明,请参见官方教程。

## 代码实现

#### 具体编码实现可参见 API 概览以及示例。

1. 创建 TCCCWorkstation 实例

```
import {TcccWorkstation,TcccErrorCode} from "tccc-sdk-uniapp";
const tcccSDK = TcccWorkstation.sharedInstance();
// 监听错误事件
tcccSDK.on("onError",(errCode,errMsg) => {
```

});

#### 2. 登录。

```
const type = TCCCLoginType.Agent;
// 其中sdkAppId、userId、token的获取参考关键概念对应的字段。
// 坐席登录
tcccSDK.login({
    sdkAppID: 140000000, // 请替换为自己的SdkAppId
    userId: "xxx@qq.com", // 替换为座席账号
    token: "xxxx", // 请替换为用调用云API接口 CreateSDKLoginToken 获取的token
    type: type,
},(code,message) => {
    if (code == TcccErrorCode.ERR_NONE) {
        // 登录成功
    } else {
        // 登录失败
    }
});
```

## △ 注意:

token 的获取需要后台开发进行,需要调用云API接口CreateSDKLoginToken来获取。

```
3. 发起呼叫。
```



## });

4. 处理对端接听回调。

```
tcccSDK.on('onAccepted',(sessionId) => {
    // 对端已接听
});
```

5. 主动挂断电话

// **结束通话** ccccSDK.terminate()



## Android

最近更新时间: 2023-07-25 15:27:52

## 快速集成腾讯云联络中心 Android SDK

本文主要介绍如何快速地将腾讯云联络中心 Android SDK 集成到您的项目中,只要按照如下步骤进行配置,就可以完成 SDK 的集成工 作。

## 开发环境要求

- Android Studio 3.5+。
- Android 4.1 (SDK API 16)及以上系统。

## 集成 SDK (aar、jar)

## 手动下载(aar、jar)

目前我们暂时还未发布到 mavenCentral ,您只能手动下载 SDK 集成到工程里:

- 1. 下载最新版本 TCCC Agent SDK。
- 2. 将下载到的 aar 文件拷贝到工程的 app/libs 目录下。
- 3. 在工程根目录下的 build.gradle 中,指定本地仓库路径。



implementation fileTree(dir: "libs",includes: ['\*.aar','\*.jar'])

4. 在 app/build.gradle的defaultConfig 中,指定 App 使用的 CPU 架构。



#### 🕛 说明:

腾讯云

目前 TCCC Agent SDK 支持 armeabi、armeabi-v7a 和 arm64-v8a。

5. 在 app/src/AndroidManifest.xml 中,指定 App 不允许应用参与备份和恢复基础架构。



6. 单击 I Sync Now, 完成 TCCC Agent SDK 的集成工作。

## 配置 App 权限

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

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
</uses-permission android:name="android.permission.READ_PHONE_STATE" />
</uses-permission.BUETOTH" />
</uses-permission android:name="android.permission.READ_PHONE_STATE" />
</uses-permission android:name="android.permission.READ_PHONE_STATE" />
</uses-permission android:name="android.permission.READ_PHONE_STATE" />
</uses-permission an
```

#### 设置混淆规则

在 proguard-rules.pro 文件,将 TCCC SDK 相关类加入不混淆名单:



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

## 代码实现

具体编码实现可参考 Android SDK API



## iOS

最近更新时间: 2024-11-05 12:03:32

## 集成云联络中心 iOS Agent SDK

本文主要介绍如何快速地将腾讯云联络中心 iOS Agent SDK 集成到您的项目中,只要按照如下步骤进行配置,就可以完成 SDK 的集成工作。

## 开发环境要求

- Xcode 9.0+。
- iOS 9.0 以上的 iPhone 或者 iPad 真机。
- 项目已配置有效的开发者签名。

## 集成 SDK

## 方案1: 使用 CocoaPods

1. 安装 CocoaPods

在终端窗口中输入如下命令(需要提前在 Mac 中安装 Ruby 环境):

sudo gem install cocoapods

2. 创建 Podfile 文件

进入项目所在路径,输入以下命令行之后项目路径下会出现一个 Podfile 文件。

pod init

 编辑 Podfile 文件 根据您的项目需要编辑 Podfile 文件:

```
platform :ios, '11.0'
target 'App' do
    pod 'TCCCSDK_Ios', :podspec =>
'https://tccc.qcloud.com/assets/doc/Agent/CppSDKRelease/TCCCSDK_Ios.podspec'
end
```

- 4. 更新并安装 SDK
  - 在终端窗口中输入如下命令以更新本地库文件,并安装 SDK:

pod install

○ 或使用以下命令更新本地库版本:

pod update

pod 命令执行完后,会生成集成了 SDK 的 .xcworkspace 后缀的工程文件,双击打开即可。

#### 方案2:手动下载

腾讯云

- 1. 下载最新版本 TCCC Agent SDK。
- 2. 打开您的 Xcode 工程项目,选择要运行的 target,单击 Build Phases 项。



3. 单击 Link Binary with Libraries 项展开,单击底下的"+"号图标去添加依赖库。



4. 依次添加下载的 TCCCSDK.Framework、TXFFmpeg.xcframework、TXSoundTouch.xcframework,及其所需依赖 库 GLKit.framework、AssetsLibrary.framework、SystemConfiguration.framework、libsqlite3.0.tbd、 CoreTelephony.framework、AVFoundation.framework、OpenGLES.framework、Accelerate.framework、 MetalKit.framework、libresolv.tbd、MobileCoreServices.framework、libc++.tbd、CoreMedia.framework。





5. 单击 General,选择 Frameworks, Libraries, and Embedded Content,检查 TCCCSDK.framework所需要动态库 TXFFmpeg.xcframework、TXSoundTouch.xcframework是否已经添加,是否正确选择 Embed & Sign,如果没有单击 底下的"+"号图标依次添加。

	main	🛃 tccc-agent-ios-example 🔪 📋 iPhone 14 Pro 🛛 Build Succeeded   2	2023/6/13 at 15:18 💧 19 📣 1
■ X II Q A ◊ Ø D I	⊞   < > ontroller.mm   h 7C	CCCode.h   h GenerateTestUserToken.h   🖾 Assets.xcassets	🔀 tccc-agente.xcodeproj
V 🖪 tccc-agent-ios-example	🖾 tccc-agent-ios-example		
tccc-agent-ios-example	Genera	al Signing & Capabilities Resource Tags Info Build Settings	Build Phases Build Rules
> Debug			
m GenerateTestUserToken.m	PROJECT		
h GenerateTestUserToken.h	🚨 tccc-agent-ios-exa 💙 Fr	rameworks, Libraries, and Embedded Content	
m <sup>†</sup> ViewController.mm		Name	Embed
h AppDelegate.h	TARGETS	🚔 Accelerate.framework	Do Not Embed 🗘
M AppDelegate.m	▲ tccc-agent-ios-exa	AssetsLibrary.framework	Do Not Embed 🗘
h SceneDelegate.h		AVFoundation.framework	Do Not Embed 🗘
M SceneDelegate.m		🚘 CoreMedia.framework	Do Not Embed 🗘
h ViewController.h		🚘 CoreTelephony.framework	Do Not Embed 🗘
🗙 Main.storyboard		🚘 GLKit.framework	Do Not Embed 🗘
🖾 Assets.xcassets		,Ē libc++.tbd	
🗙 LaunchScreen.storyboard		,∄ libresolv.tbd	
🖽 Info.plist		🖉 libsqlite3.0.tbd	
m main.m		🚔 MetalKit.framework	Do Not Embed 🗘
> 🚍 Products		🚔 MobileCoreServices.framework	Do Not Embed 🗘
> 🚍 Frameworks		🕂 OpenGLES.framework	Do Not Embed 🗘
		🖴 SystemConfiguration.framework	Do Not Embed 🛇
		🚔 TCCCSDK.framework	Embed & Sign 🗘
		TXFFmpeg.xcframework	Embed & Sign 🗘
		🚔 TXSoundTouch.xcframework	Embed & Sign ≎
	✓ D:	evelopment Assets	

6. 在工程 target 中 Build Settings 的 Other Linker Flags 增加 - ObjC 配置。



	P tccc-agent-ios-examin	mple	Kccc-agent-ios-ex	ample 🔪 📋 iPhone 14	4 Pro Build Succeeded	2023/6/13 at 15:18 💧 19 斗 1
╘ 🛛 🖓 ۹ ۵ 🖉 🗖	$\mathbb{B}$   $<$ $>$ ontroller.mm	h <i>tccc</i>	Code.h   h Gene	erateTestUserToken.h	Assets.xcassets	s 🔣 tccc-agente.xcodepro
✓	🛃 tccc-agent-ios-example					
Tecc-agent-ios-example     Tecc-agent-io		General	Signing & Capabilitie	s Resource Tags	Info Build Settings	Euild Phases Build Rules
~ 🚞 Debug		+ Basi	c Customized All	Combined Lev	rels	📄 🕞 other link
M GenerateTestUserToken.m	PROJECT					
h GenerateTestUserToken.h	🛃 tccc-agent-ios-exa	Linking				
m <sup>*</sup> ViewController.mm		↓ Linking			A toos-agen	t-ios-example
h AppDelegate.h	TARGETS		Other Linker Flags		-ObiC	
m AppDelegate.m	A tccc-agent-ios-exa		Quote Linker Argumer	nts	Yes ¢	
h SceneDelegate.h						
M SceneDelegate.m						
h ViewController.h						
🗙 Main.storyboard						
🖾 Assets.xcassets						
🗙 LaunchScreen.storyboard						
🖽 Info.plist						
m main.m						
> 🔚 Products						

## 配置 App 权限

1. 如需使用 SDK 提供的音视频功能,需要给 App 授权麦克风的使用权限。在 App 的 Info.plist 中添加对应麦克风在系统弹出授权对 话框时的提示信息。



2. 如需 App 进入后台仍然运行相关功能,可在 XCode 中选中当前工程项目,并在 Capabilities 下将设置项 Background Modes 设定为 ON,并勾选 Audio, AirPlay and Picture in Picture,如下图所示:





## 代码实现

目前我们提供了 Swift、OC、C++ 接口供开发者选择使用,可以用下面代码引入头文件:

<pre>import TCCCSDK // 获取tcccSDK 单例 let tcccSDK: TCCCWorkstation = {    return TCCCWorkstation.sharedInstance() }() // 获取SDK版本号 let version = TCCCWorkstation.getSDKVersion()</pre>	Swift	
	<pre>import TCCCSDK // 获取tcccSDK 单例 let tcccSDK: TCCCWorkstation = { return TCCCWorkstation.sharedInstance() }() // 获取SDK版本号 let version = TCCCWorkstation.getSDKVersion()</pre>	

#### Objective-C





#### NSString\* version = [TCCCWorkstation getSDKVersion];

#### C++

```
// 引入C++头文件
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
// 使用tccc命名空间
using namespace tccc;
// 获取tcccSDK 单例
ITCCCWorkstation* tcccSDK = getTCCCShareInstance();
// 获取SDK版本号
const char * version = tcccSDK->getSDKVersion();
```

具体编码实现可参见 API 概览以及示例。

## 常见问题

## 如何查看 TCCC 日志?

TCCC 的日志默认压缩加密,后缀为 .log。

• iOS 日志路径: sandbox/Documents/tccc

## 在 iOS 下回调是否都在主线程

Swift、OC 接口的所有回调均在主线程,开发者无需特别处理。但 C++ 接口下回调都不在主线程,需要业务层面上判断并且把他转为主 线线程:

```
if ([NSThread isMainThread]) {
    // 在主线程,直接可以处理
    return;
}
dispatch_async(dispatch_get_main_queue(), ^{
    // 回调在非主线程。
});
```



## 工作台 SDK API 文档 Web

最近更新时间: 2025-06-30 21:49:02

#### ▲ 注意:

TCCC 是加载 SDK 后的全局变量,可直接访问。

## 通用结构

## AgentStatus

座席状态。

字段	描述
free	空闲
busy	忙碌
arrange	话后整理
notReady	示忙
rest	小休

## ServerType

端服务类型,描述电话类型会话时使用的端类型。

字段	描述
staffSeat	Web 座席类型
staffPhoneSeat	座席手机类型
miniProgramSeat	小程序类型
staffExtensionSeat	话机类型

## CommonSDKResponse

参数		类型	必填	备注
options	status	'success'  'error'	是	SDK API 调用结果,成功时返回 success,失败返回 error
	errorMsg	string	否	错误信息,当 status 为 error 时返回

## Call(电话客服和音频客服相关接口函数)

## 电话呼出



## tccc.Call.startOutboundCall(options): Promise<CallResponse>

参数		类型	必填	备注
phoneNum phoneDesc uui	phoneNumber	Strin g	是	被叫号码
	phoneDesc	Strin g	否	号码备注,在通话条中会替代号码显示
	uui	Strin g	否	用户自定义数据,传入后可通过 电话 CDR 事件 推送返回
option s	skillGroupId	Strin g	否	指定技能组内绑定的外呼号码
	callerPhoneNum ber	Strin g	否	指定外呼号码
	servingNumberGr oupIds	Strin g∏	否	指定号码 ID 列表
	phoneEncodeTyp e	'num ber'	否	目前仅支持 'number',在开启 <mark>号码映射</mark> 时强制使用真实号码

### CallResponse 描述如下:

参数		类型	必填	备注
session callee callee mber	sessionId	Strin g	是	指定会话 ID
	calleeLocation	Strin g	否	被叫号码归属地址
	calleePhoneNu mber	Strin g	是	被叫号码
se	callerPhoneNum ber	Strin g	是	外呼时使用的主叫号码
	serverType	Strin g	是	表示外呼时使用的端类型,可选值有:staffSeat, staffPhoneSeat,staffExtensionSeat。详细说明参见 <mark>会话</mark> 服务类型
	remark	Strin g	否	被叫号码备注

## 接待会话

## tccc.Call.accept(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID,从 tccc.events.callIn 事件中获



		取

## 挂断会话

## tccc.Call.hungUp(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

### 删除会话

## tccc.Call.deleteCall(options)

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

静音

## tccc.Call.muteMic(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

## 取消静音

## tccc.Call.unmuteMic(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

## 当前是否静音

### tccc.Call.isMicMuted(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 发起内部通话

## tccc.Call.startInternalCall(): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	calleeUserId	String	是	被叫座席账号
	useMobile	Boolean	否	是否呼叫对方手机

## 转接会话

## tccc.Call.transfer(): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID
	skillGroupId	String	否	转接到指定技能组
	userld	String	否	转接到指定座席
	phone	String	否	转接到指定号码

## 呼叫保持

## tccc.Call.hold(): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

## 通话监听

## tccc.Call.monitor(options): Promise<CommonSDKResponse>

发起通话监听,默认是语音监听,只能同时发起一个。当指定 textOnly: true 时,可同时发起多个文字监听,该参数需开启 <mark>实时语音转</mark> 文字 功能

注意该 API 需管理员或质检员角色才能调用。

参数		类型	必填	备注
options	sessionId	String	是	被监听的会话 ID,可从 会 话信息列表 中获取
	textOnly	Boolean	否	默认为 false,表示发起语 音监听,指定为 true 表示发 起文字监听

## 强拆

## tccc.Call.intercept(options): Promise<CommonSDKResponse>

发起通话强拆,被强拆的会话必须处于被监听状态。

参数		类型	必填	备注
options	sessionId	String	是	被强拆的会话ID

## 取消通话保持

## tccc.Call.unHold(): Promise<CommonSDKResponse>

参数	类型	必填	备注
----	----	----	----


# 发送分机号

# tccc.Call.sendDigits(): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID
	dtmfText	String	否	需要发送的分机号

# 开启ASR(语音识别功能)

#### tccc.Call.startASR(options): Promise<CommonSDKResponse>

#### 使用该方法需要购买语音识别套餐包

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

# 关闭ASR(语音识别功能)

#### tccc.Call.stopASR(): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

# Chat (在线客服相关接口函数)

# 接听会话

#### tccc.Chat.accept(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

## 结束会话

# tccc.Chat.end(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 转接会话

# tccc.Chat.transfer(): Promise<CommonSDKResponse>

参数	类型	必填	备注
----	----	----	----



options	sessionId	String	是	指定会话 ID
	skillGroupId	String	否	转接到指定技能组
	userld	String	否	转接到指定座席

# Video(视频客服相关接口函数)

# 接听会话

#### tccc.Video.accept(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 挂断会话

#### tccc.Video.end(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 静音

#### tccc.Video.muteMic(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 取消静音

#### tccc.Video.unmuteMic(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 关闭摄像头

#### tccc.Video.muteVideo(options): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 开启摄像头

tccc.Video.unmuteVideo(options): Promise<CommonSDKResponse>



参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID

#### 转接会话

#### tccc.Video.transfer(): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID
	skillGroupId	String	否	转接到指定技能组
	userld	String	否	转接到指定座席

# Agent (座席状态相关接口函数)

更多座席状态枚举类型请参见 座席状态。

#### 上线

tccc.Agent.online(): void

#### 下线

tccc.Agent.offline(): void

#### 设置座席状态

#### tccc.Agent.setStatus(optoins): Promise<CommonSDKResponse>

参数		类型	必填	备注
options	status	String	是	座席状态,可选值: • free:空闲 • rest:小休 • notReady:示忙 • stopNotReady:停止示忙
	restReason	String	否	小休原因

#### 获取座席状态

tccc.Agent.getStatus():AgentStatus

# Devices(设备相关接口函数)

#### 检测当前浏览器是否支持

#### tccc.Devices.isBrowserSupported(): boolean

```
() 说明:
```

TCCC Web SDK 支持 Chrome 56、Edge 80以上的浏览器。



tccc.Devices.getMicrophones(): Promise<MediaDeviceInfo []>

#### 返回扬声器设备列表

tccc.Devices.getSpeakers(): Promise<MediaDeviceInfo []>

# UI(座席界面相关接口函数)

#### 隐藏 SDK 所有 UI

tccc.UI.hide(): void

#### 显示 SDK 所有 UI

tccc.UI.show(): void

#### 显示浮动按钮

tccc.UI.showfloatButton(): void

#### 隐藏浮动按钮

tccc.UI.hidefloatButton(): void

显示工作台

tccc.UI.showWorkbench(): void

#### 隐藏工作台

tccc.UI.hideWorkbench(): void

#### 显示通话条

tccc.UI.showNotificationBar(): void

#### 隐藏通话条

tccc.UI.hideNotificationBar(): void

#### 修改SDK本地设置

支持关闭SDK铃声和系统通知

#### tccc.UI.updateUserCustomSettings(settings): void

settings内参数都是可选项,支持增量更新。

参数		类型	必填	备注
settings disableRingtone		Boolean	true 表示禁用 SDK 的铃 否 声,包括来电铃声、接听铃 声	
	disableNotificatio	Boolean	否	true 表示禁用 SDK 的系统



n		通知

# Events (事件)

## 事件监听

tccc.on(event, callback)

# 取消事件监听

tccc.off(event, callback)

# SDK 初始化完成

#### tccc.events.ready

当 SDK 初始化完成时触发,此时可安全调用 API。

callback 参数		类型	必填	备注
options	tabUUID	String	是	表示当前页面的唯一 ID,刷 新后会变,用于多 Tab 集成 SDK

#### 会话呼入

#### tccc.events.callIn

会话呼入类型包括:

- phone: 电话会话
- im: 在线会话
- voip: 音频会话
- video:视频会话
- internal: 内线会话

#### 电话会话呼入

callback	参数	类型	必填	备注
option	sessionId	String	是	会话 ID
S	type	'phone'	是	电话会话类型
	timeout	Number	是	会话接入超时时长,0代表不超时
	calleePhoneN umber	String	是	被叫号码
	callerPhoneN umber	String	否	主叫号码
	callerLocation	String	否	主叫号码归属地
	remark	String	否	备注



ivrPath	{key: String, label: String}[]	-	用户的 IVR 按键路径,key 表示对应按键,label 表示对应的按键标签
protectedCall ee	String	否	在开启号码映射时存在,表示被叫
protectedCall er	String	否	在开启号码映射时存在,表示主叫
serverType	'staffSeat'   'staffPhoneSeat'   'staffExtensionSeat'	是	表示呼入到座席哪一端,staffSeat 为默认值,表 示 Web 座席; StaffPhoneSeat 表示呼入到座席 手机,MiniProgramSeat 表示小程序座席, staffExtensionSeat 表示呼入到座席绑定的话机

# 在线会话呼入

callback 参数		类型	必填	备注
	sessionId	String	是	会话 ID
	type	'phone'	是	电话会话类型
	timeout	Number	是	会话接入超时时长,0代表不超时
	nickname	String	是	用户昵称
	avatar	String	否	用户头像
option S	remark	String	否	备注
	peerSour ce	String	否	渠道来源
	channelN ame	String	否	自定义参数
	clientDat a	String	否	用户自定义参数

# 音频会话呼入

callback	参数	类型	必填	备注
option	sessionId	String	是	会话 ID
S	type	'voip'	是	音频会话类型
	timeout	Number	是	会话接入超时时长,0代表不超时
	callee	String	是	渠道入口
	calleeRe mark	String	否	渠道入口备注
	userld	String	是	用户的 openId



nickname	String	否	用户授权后可获得微信昵称
avatar	String	否	用户授权后可获得微信头像
remark	String	否	备注
peerSour ce	String	否	主叫号码归属地
ivrPath	{key: String, label: String}[]	否	用户的 IVR 按键路径,key 表示对应按键,label 表示对应 的按键标签
clientDat a	String	否	用户自定义参数

# 视频会话呼入

callback	参数	类型	必填	备注
	sessio nId	String	是	会话 ID
	type	'video'	是	视频会话类型
	timeo ut	String	是	会话接入超时时长,0代表不超时
option s	userld	String	是	用户的 openId
	nickna me	String	否	用户授权后可获得微信昵称
	avatar	String	否	用户授权后可获得微信头像
	remar k	String	否	备注

# 内部会话呼入

callback 参数		类型	必填	备注
option	session Id	String	是	会话 ID
	type	'internal'	是	内部会话类型
S	timeout	Number	是	会话接入超时时长,0代表不超时
	peerUs erId	String	是	主叫座席的账号

# 座席接入会话

#### tccc.events.userAccessed

备注	注	备注
----	---	----



option	sessio nId	String	是	指定会话 ID
S	tabUU ID	String	否	开启多 Tab 集成时存在,表示哪个 Tab 接听的会话

# 会话超时转接事件

#### tccc.events.autoTransfer

callback	参数	类型	必填	备注
option s	sessio nId	String	是	指定会话 ID

# 会话结束事件

#### tccc.events.sessionEnded

callback 参数		类型	必填	备注
option s	sessionl d	String	是	指定会话 ID
	closeBy	String	是	表示挂断方: • user:用户挂断 • seat:座席挂断 • admin:系统挂断 • timer: 定时器挂断
	mainRea son	String	否	仅在电话类型,并且挂断方为"admin"时存在,表示挂断原因
	subReas on	String	否	仅在电话类型,并且挂断方为"admin"时存在,表示挂断的详细原因

# 外呼成功事件

#### tccc.events.callOuted

callback 参数		类型	必填	备注
option	sessionId	String	是	指定会话 ID
S	callerPhoneNum ber	String	是	外呼使用的主叫号码
	calleePhoneNu mber	String	是	被叫号码
	serverType	'staffSeat'   'staffPhoneSeat'   'staffExtensionSeat'   'MiniProgramSeat'	是	表示座席外呼类型: <ul> <li>staffSeat 为默认值,表示 Web 座席</li> <li>StaffPhoneSeat 表示使用手机外呼</li> <li>MiniProgramSeat 表示使用小程序外</li> </ul>



			● staffExtensionSeat 表示使用话机外呼
tabUUID	String	否	开启多Tab集成时存在,表示哪个 Tab 发起 的外呼

#### 外呼对方接听事件

#### tccc.events.calloutAccepted

callback	参数	类型	必填	备注
option s	sessio nId	String	是	指定会话 ID

# 会话转接事件

#### tccc.events.transfer

callback	参数	类型	必填	备注
option s	sessio nId	String	是	指定会话 ID

# 座席状态变更事件

# tccc.events.statusChanged

callback	参数	类型	必填	备注
option s	status	AgentStatu s	否	详细说明请参见 座席状态

#### 座席被踢下线事件

#### tccc.events.kickedOut

座席多端登录时触发。

#### 语音识别事件

#### tccc.events.asr

callback	参数	类型	必填	备注
option res	sessio nId	String	是	指定会话 ID
	result	ASR 识别结果	是	语音识别结果,结构体请参见 文档
	flow	'IN'   'OUT'	是	识别方向。 ● IN:用户侧 ● OUT:座席侧

# 初始化参数



#### TCC Web SDK允许在初始化时,新增dataset参数用于开启功能特性 使用方法如下:

```
function injectTcccWebSDK(SdkURL) {
    if (window.tccc) {
        console.warn('已经初始化SDK了,请确认是否重复执行初始化');
        return;
    }
    return new Promise((resolve, reject) => {
        const script = document.createElement('script');
        script.setAttribute('crossorigin', 'anonymous');
        script.src = SdkURL;
        /*
        * 增加dataset参数,表示开启/关闭功能特性
        */
        script.dataset.xxx = 'true'
        document.body.appendChild(script);
    })
}
```

#### 相关 dataset 如下表:

dataset 参数	备注
enableShared	表示启用 多 Tab 功能 。
disableNotification	关闭浏览器通知。
disableRingtone	关闭 SDK 所有铃声。
disableCDR	关闭 SDK 工作台服务记录展示。
disableUI	不初始化 SDK 所有 UI,包括工作台和通话条。
disablePreloadMic	登录后不启动麦克风,并且每次通话完成后释放麦克风。

# 多 Tab 集成 SDK

默认情况下,TCCC Web SDK 只允许在一个地方登录,多处登录会触发 kickedOut 事件。开启多 Tab 功能后,任意一个页面发起 的通话,都会在其他页面显示,开发者可根据业务逻辑自行隐藏 UI,或者监听对应事件处理。

#### 限制条件

- 1. 同一个浏览器的多个窗口,注意不能开启无痕模式。
- 2. SDK 集成在业务系统处于同一个域名下。
- 3. 不支持移动端浏览器。

#### 集成步骤

- 1. 初始化 SDK,参考 Web。
- 2. 增加 enableShared 参数,表示启用多 Tab 功能。



```
function injectTcccWebSDK(SdkURL) {
    if (window.tccc) {
        console.warn('已经初始化SDK7, 请确认是否重复执行初始化');
        return;
    }
    return new Promise((resolve, reject) => {
        const script = document.createElement('script');
        script.setAttribute('crossorigin', 'anonymous');
        script.src = SdkURL;
        /*
        * 增加enableShared, 表示启用多Tab功能
        */
        script.dataset.enableShared = 'true'
        document.body.appendChild(script);
        script.addEventListener('load', () => {
            window.tccc.on(window.tccc.events.ready, ({ tabUUID }) => {
            resolve('初始化成功, 当前tabUUID为' + tabUUID)
        });
        window.tccc.on(window.tccc.events.tokenExpired, ({message}) => {
            console.error('初始化失败', message)
            reject(message)
        })
    })
    })
}
```

#### 3. 处理多 Tab 逻辑。

触发 callOuted (外呼成功)和 userAccessed (座席接听成功)事件时,会增加 tabUUID 字段,表示哪个页面发起的外呼/接听。

```
let curTabUUID = '';
window.tccc.on(window.tccc.events.ready, ({ tabUUID }) => {
    console.log('初始化成功, 当前tabUUID为' + tabUUID)
    curTabUUID = tabUUID;
});
window.tccc.on(window.tccc.events.callouted, ({ sessionId, tabUUID }) => {
    if (tabUUID && tabUUID !== curTabUUID) {
        // 接收到其他页面的外呼成功事件, 业务可自行处理
    }
})
window.tccc.on(window.tccc.events.userAccessed, ({ sessionId, tabUUID }) => {
    if (tabUUID && tabUUID !== curTabUUID) {
        // 接收到其他页面的按听成功事件, 业务可自行处理
        // 接收到其他页面的接听成功事件, 业务可自行处理
        // 此处为示例代码, 会忽略该事件
        return;
    }
})
```





# uni-app

最近更新时间: 2024-09-13 17:47:01

# API 概览

#### 创建实例和事件回调

API	描述
sharedInstance	创建 TCCCWorkstation 实例(单例模式)
destroyInstance	销毁 TCCCWorkstation 实例(单例模式),建议您在不使用 tccc 的时候卸载 tccc 实例
on	设置 TCCCWorkstation 事件回调
off	取消 TCCCWorkstation 事件回调

#### 创建实例和设置事件回调示例代码

```
// 引入TCCC相关包
import {TcccWorkstation,TCCCLoginType,TCCCAudioRoute,TCCCEndReason} from "tccc-sdk-
uniapp";
// 创建实例和设置事件回调
const tcccSDK = TCCCWorkstation.sharedInstance();
// 错误事件回调
tcccSDK.on('onError',(errCode,errMsg) => {
});
// 通话结束回调
tcccSDK.on('onEnded',(reason,reasonMessage,sessionId) => {
    if (reason == TCCCEndReason.Error) {
        // 呼叫异常
    }
});
// 对端接听回调
tcccSDK.on('onAccepted',(sessionId) => {
}// 释放所有事件回调监听
tcccSDK.of('*');
```

#### 登录相关接口函数

API	描述
login	SDK 登录
checkLogin	检查 SDK 登录状态,建议您在页面 onShow 的时候调用
logout	SDK 退出登录



#### 登录示例代码

```
// 其中sdkAppId、userId、token的获取参考关键概念对应的字段。
// 座席登录
// 手机应用程序在切换到后台时,操作系统会暂停应用程序的进程以节省资源。我们建议您在 onShow 的时候做一个
     // 未登录
```

# 呼叫相关接口函数

API	描述
call	发起通话
answer	接听来电
terminate	结束通话
sendDTMF	发送 DTMF(双音多频信号)
mute	静音
unmute	取消静音
startPlayMusic	开始播放音乐



stopPlayMusic

停止播放音乐

#### 发起呼叫和结束呼叫示例代码

// 发起呼叫	
to: '134xxxx',	// <b>被叫号码(必填)</b>
remark: "xxx",	// 号码备注,在通话条中会替代号码显示(可选)
uui: "xxxx",	// 户自定义数据(可选)
}, (code,message) => {	
if (code == TcccError	Code.ERR_NONE) {
// 发起成功	
// 发起失败	
// 结束通话	
<pre>tcccSDK.terminate();</pre>	
// 接听来听	
<pre>tcccSDK.answer((code,me</pre>	ssage) => {
if (code == TcccErr	orCode.ERR_NONE) {
// 接听成功	
// 接听失败	

#### 音频设备接口函数

API	描述
setAudioCaptureVolume	设定本地音频的采集音量
getAudioCaptureVolume	获取本地音频的采集音量
setAudioPlayoutVolume	设定远端音频的播放音量
getAudioPlayoutVolume	获取远端音频的播放音量
setAudioRoute	设置音频路由

```
// TCCCAudioRoute.Earpiece 为耳麦
// 设置为扬声器
const route = TCCCAudioRoute.Speakerphone;
tcccSDK.getDeviceManager().setAudioRoute(route);
```

# 调试相关接口

API

描述



getSDKVersion	获取 SDK 版本信息
setLogLevel	设置 Log 输出级别
setConsoleEnabled	启用/禁用控制台日志打印

#### 获取SDK版本示例代码

// **获取**SDK 版本号

TCCCWorkstation.getSDKVersion();

#### 错误和警告事件

API	描述
onError	错误事件回调
onWarning	警告事件回调

#### 处理错误回调事件回调示例代码

// 错误事件回调	
<pre>tcccSDK.on('onError', (errCode, errMsg) =&gt; {</pre>	
// 警告事件回调	
<pre>tcccSDK.on('onWarning',(warningCode,warningMsg) =&gt; {</pre>	

#### 呼叫相关事件回调

API	描述
onNewSession	新会话事件。包括呼入和呼出
onAccepted	对端接听回调
onEnded	会话结束事件
onAudioVolume	音量大小的反馈回调
onNetworkQuality	网络质量的实时统计回调

#### 处理接听和座席挂断事件回调示例代码





```
msg = "您已挂断";
      msg = "对方已挂断";
      msg = "对方已拒接";
      msg = "对方已取消";
// 新会话事件。包括呼入和呼出
      // 呼入,因手机切后台的时候是不能收到该事件的。所以这里建议您开通手机接听的能力
   ///当前网络一般
   ///当前网络很差
   ///当前网络不满足 通话 的最低要求
// 音量大小的反馈回调.volume从0到100,数值越大表示声音越大。
tcccSDK.on('onAudioVolume', (userId, volume) => {
```

}),

#### 与云端连接情况的事件回调

API	描述
onConnectionLost	SDK 与云端的连接已经断开
onTryToReconnect	SDK 正在尝试重新连接到云端
onConnectionRecovery	SDK 与云端的连接已经恢复

#### 与云端连接情况的事件回调示例代码

tcccSDK.on('onConnectionLost', (serverType) => {



#### // 与云端的连接已经断开

```
});
tcccSDK.on('onTryToReconnect',(serverType) => {
    // 正在尝试重新连接到云端
});
tcccSDK.on('onConnectionRecovery',(serverType) =>
    // 与云端的连接已经恢复
}):
```

# API 错误码

#### 基础错误码

符号	值	含义
ERR_NONE	0	无错误。成功
ERR_HTTP_REQUEST_FAILURE	-10001	Http 请求失败,请检查网络连接情况
ERR_HTTP_TOKEN_ERROR	-10002	token 登录票据不正确或者已过期
ERR_HTTP_GETSIPINFO_ERROR	-10003	获取座席配置失败。请联系我们
ERR_NETWORK_CANNOT_RESET	-10004	正在通话中,禁止重置网络操作&发起外呼
ERR_HAD_LOGGEDOUT	-10005	您已经退出登录了,请重新登录
ERR_UNRIGIST_FAILURE	20001	注销失败
ERR_ANSWER_FAILURE	20002	接听失败,通常是 trtc 进房失败
ERR_SIPURI_WRONGFORMAT	20003	URI 格式错误。

# SIP 相关错误码

符号	值	含义
ERR_SIP_BAD_REQUEST	400	错误请求。通常是座席没有登录就发起了请求
ERR_SIP_UNAUTHORIZED	401	未授权(用户名密码不对情况)
ERR_SIP_PAYMENTREQUIRED	402	付费要求,通常是座席许可满了
ERR_SIP_FORBIDDEN	403	密码错误,或者是被踢了
ERR_SIP_REQUESTTIMEOUT	408	请求超时(网络超时)
ERR_SIP_REQUEST_TERMINATED	487	请求终止(网络异常,网络中断场景下)
ERR_SIP_SERVICE_UNAVAILABLE	503	服务不可用
ERR_SIP_SERVER_TIMEOUT	504	服务超时

#### 音频设备相关错误码



符号	值	含义
ERR_MIC_START_FAIL	-1302	打开麦克风失败。设备,麦克风的配置程序(驱动程序)异 常,禁用后重新启用设备,或者重启机器,或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权,通常在移动设备出现,可能是权限被用户 拒绝了
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中,例如移动设备正在通话时,打开麦克风 会失败
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败,例如在 Windows 或 Mac
ERR_SPEAKER_SET_PARAM_FAIL	-1322	扬声器设置参数失败
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败
ERR_UNSUPPORTED_SAMPLERAT E	-1306	不支持的音频采样率

# 网络相关错误码

符号	值	含义
ERR_RTC_ENTER_ROOM_FAI LED	-3301	进入房间失败,请查看 onError 中的 −3301 对应的 msg 提示确认失 败原因
ERR_RTC_REQUEST_IP_TIME OUT	-3307	请求 IP 和 sig 超时,请检查网络是否正常,或网络防火墙是否放行 UDP
ERR_RTC_CONNECT_SERVER _TIMEOUT	-3308	请求进房超时,请检查是否断网或者是否开启vpn,您也可以切换4G进 行测试确认
ERR_RTC_ENTER_ROOM_REF USED	-3340	进房请求被拒绝,请检查是否连续调用 enterRoom 进入相同 ID 的房 间



# Android

最近更新时间: 2025-01-16 11:43:42

#### 创建实例和事件回调

API	描述
sharedInstance	创建 TCCCWorkstation 实例(单例模式)。
destroySharedInstance	销毁 TCCCWorkstation 实例(单例模式)。
setListener	设置 TCCCWorkstation 事件回调。

#### 创建实例和设置事件回调示例代码

// 创建实例和设置事件回调
TCCCWorkstation tcccSDK = TCCCWorkstation.sharedInstance(getApplicationContext());
tcccSDK.setListener(new TCCCListener() {});

#### 登录相关接口函数

API	描述
login	SDK 登录。
checkLogin	检查 SDK 是否已登录。
logout	SDK 退出登录。

#### 登录示例代码

```
TCCCTypeDef.TCCCLoginParams loginParams = new TCCCTypeDef.TCCCLoginParams();
/// 登录的坐席ID,通常为邮箱地址
loginParams.userId = "";
/// 登录票据,在登录模式为Agent必填。更多详情请参见[创建 SDK 登录
/// Token](https://cloud.tencent.com/document/product/679/49227)
loginParams.token = "";
/// 腾讯云联络中心应用ID,通常为1400开头
loginParams.sdkAppId = 0;
// 必须知道为坐席模式
loginParams.type = TCCCTypeDef.TCCCLoginType.Agent;
tcccSDK.login(loginParams, new TXCallback() {
    @Override
    public void onSuccess() {
        // login success
        }
    @Override
    public void onError(int code, String desc) {
```



});

#### 呼叫相关接口函数

API	描述
call	发起通话。
answer	接听来电。
terminate	结束通话。
sendDTMF	发送 DTMF(双音多频信号)。
mute	静音。
unmute	取消静音。

#### 发起呼叫和结束呼叫示例代码

```
TCCCTypeDef.TCCCStartCallParams callParams =new TCCCTypeDef.TCCCStartCallParams();
//#dx <scheme> : <user> @<host>, 如 sip:1343xxxx@1400xxxx.tccc.qcloud.com, 其中1343xxxx/b
FMG, 1400xxxx/b@btccc@mID
callParams.to = "sip:1343xxxx@1400xxxx.tccc.qcloud.com";
// 发起通话
tcccSDK.call(callParams, new TXCallback() {
    @Override
    public void onSuccess() {
        // call success
     }
    @Override
    public void onError(int code, String desc) {
        // call error
        }
    });
    // datpata
tcccSDK.terminate();
```

#### 音频设备接口函数

API	描述
setAudioCaptureVolume	设定本地音频的采集音量。
getAudioCaptureVolume	获取本地音频的采集音量。
setAudioPlayoutVolume	设定远端音频的播放音量。
getAudioPlayoutVolume	获取远端音频的播放音量。



#### setAudioRoute

设置音频路由。

# 调试相关接口

API	描述
getSDKVersion	获取 SDK 版本信息。
setLogLevel	设置 Log 输出级别。
setConsoleEnabled	启用/禁用控制台日志打印。
callExperimentalAPI	调用实验性接口。

# 获取SDK版本示例代码

// <b>获取</b> SDK <b>版本号</b>	

# 错误和警告事件

API	描述
onError	错误事件回调。
onWarning	警告事件回调。

#### 处理错误回调事件回调示例代码

	* 错误事件,表示 SDK 抛出的不可恢复的错误,比如进入房间失败或设备开启失败等。
	* @param errCode <b>错误码</b>
	* @param errMsg <b>错误信息</b>
	* @param extraInfo <b>扩展信息字段,个别错误码可能会带额外的信息帮助定位问题</b>
00v	erride
pub	lic void onError(int errCode, String errMsg, Bundle extraInfo) {
	<pre>super.onError(errCode, errMsg, extraInfo);</pre>
	* 警告事件回调
	* 警告事件,表示 SDK 抛出的提示性问题,比如音频出现卡顿或 CPU 使用率太高等。
	* @param warningCode <b>警告码</b>
	* @param warningMsg <b>警告信息</b>



super.onWarning(warningCode, warningMsg, extraInfo)

#### });

#### 呼叫相关事件回调

API	描述
onNewSession	新会话事件。包括呼入和呼出。
onEnded	会话结束事件。
onAudioVolume	音量大小的反馈回调。
onNetworkQuality	网络质量的实时统计回调。

#### 处理接听和坐席挂断事件回调示例代码

```
tcccSDK.setListener(new TCCCListener() {
    @Override
    public void onNewSession(TCCCTypeDef.ITCCCSessionInfo info) {
        super.onNewSession(info);
        // 新会活事件。包括呼入和呼出,可通过 info.sessionDirection 判断是呼入还是呼出
    }
    @Override
    public void onEnded(int reason, String reasonMessage, String sessionId) {
        super.onEnded(reason, reasonMessage, sessionId);
        // 会话结束
    }
    @Override
    public void onAccepted(String sessionId) {
        super.onAccepted(sessionId);
        // 对端接听
    }
});
```

#### 与云端连接情况的事件回调

API	描述
onConnectionLost	SDK 与云端的连接已经断开。
onTryToReconnect	SDK 正在尝试重新连接到云端。
onConnectionRecovery	SDK 与云端的连接已经恢复。

#### 与云端连接情况的事件回调示例代码

```
tcccSDK.setListener(new TCCCListener() {
    /**
```



```
* SDK 会在跟云端的连接断开时抛出此事件回调,导致断开的原因大多是网络不可用或者网络切换所致,
* 比如用户在通话中走进电梯时就可能会遇到此事件。 在抛出此事件之后,SDK 会努力跟云端重新建立连
* 所以, SDK 会在如下三个连接相关的事件中按如下规律切换:
* SDK 正在尝试重新连接到云端
* 连接恢复后会抛出 onConnectionRecovery。
* SDK 会在跟云端的连接断开时抛出 onConnectionLost,之后会努力跟云端重新建立连接并抛出
* 连接恢复后会抛出本事件回调。
```

# API 错误码

#### 基础错误码

符号	值	含义
ERR_SIP_SUCCESS	200	成功。
ERR_UNRIGIST_FAILURE	20001	登录失败。
ERR_ANSWER_FAILURE	20002	接听失败,通常是 TRTC 进房失败。
ERR_SIPURI_WRONGFORMAT	20003	URI 格式错误。
ERR_HTTP_REQUEST_FAILURE	-10001	HTTP 请求失败,请检查网络连接情况。



ERR_HTTP_TOKEN_ERROR	-10002	token 登录票据不正确或者已过期。
ERR_HTTP_GETSIPINFO_ERROR	-10003	获取坐席配置失败,请 联系 <mark>我们</mark> 。

# SIP相关错误码

符号	值	含义
ERR_SIP_BAD_REQUEST	400	错误请求。
ERR_SIP_UNAUTHORIZED	401	未授权(用户名密码不对情况)。
ERR_SIP_PAYMENTREQUIRED	402	许可不足,需要增加或者购买坐席许可。
ERR_SIP_AUTHENTICATION_REQU	407	代理需要认证,请检查是否已经调用登录接口。
ERR_SIP_REQUESTTIMEOUT	408	请求超时(网络超时)。
ERR_SIP_REQUEST_TERMINATED	487	请求终止(网络异常,网络中断场景下)。
ERR_SIP_SERVICE_UNAVAILABLE	503	服务不可用。
ERR_SIP_SERVER_TIMEOUT	504	服务超时。

# 音频设备相关错误码

符号	值	含义
ERR_MIC_START_FAIL	-1302	打开麦克风失败。设备,麦克风的配置程序(驱动程序)异 常,禁用后重新启用设备,或者重启机器,或者更新配置程 序。
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权,通常在移动设备出现,可能是权限被用户 拒绝了。
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败。
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中,例如移动设备正在通话时,打开麦克风 会失败。
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败。
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败,例如在 Windows 或 Mac。
ERR_SPEAKER_SET_PARAM_FAIL	-1322	扬声器设置参数失败。
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败。
ERR_UNSUPPORTED_SAMPLERAT E	-1306	不支持的音频采样率。

# 网络相关错误码

符号	值	含义



ERR_RTC_ENTER_ROOM_FAILED	-3301	进入房间失败,请查看 onError 中的 −3301 对应的 msg 提 示确认失败原因。
ERR_RTC_REQUEST_IP_TIMEOUT	-3307	请求 IP 和 Sig 超时,请检查网络是否正常,或网络防火墙是否 放行 UDP。
ERR_RTC_CONNECT_SERVER_TIM EOUT	-3308	请求进房超时,请检查是否断网或者是否开启 VPN,您也可以 切换 4G 进行测试确认。
ERR_RTC_ENTER_ROOM_REFUSE D	-3340	进房请求被拒绝,请检查是否连续调用 enterRoom 进入相同 ID 的房间。



# iOS

最近更新时间: 2025-01-21 11:27:12

本文主要介绍云联络中心(TCCC)坐席端的常用 API,在 iOS 端我们提供了Swift、Objective−C、C++ 接口供开发者选择使用。 我们推荐 iOS 开发者在开发应用时候请用 Swift 语言开发。

#### 创建实例和事件回调

API	描述
sharedInstance	创建 ITCCCWorkstation 实例(单例模式)。
destroySharedInstance	销毁 ITCCCWorkstation 实例(单例模式)。
addTcccListener	添加 ITCCCWorkstation 事件回调。
removeTCCCListener	移除 ITCCCWorkstation 事件回调。

#### 创建实例和设置事件回调示例代码

Swift

```
import receible
let tcccSDK: TCCCWorkstation = {
    // 创建实例
    return TCCCWorkstation.sharedInstance
}()
// 设置TCCC事件回调
tcccSDK.addTcccListener(self)
// 移除TCCC事件回调
tcccSDK.removeTCCCListener(self)
// 銷盼实例
```

```
// 市式大切
TCCCWorkstation.destrovSharedIntance()
```

#### Objective-C





```
// 创建实例
__tcccSDK = [TCCCWorkstation sharedInstance];
}
return _tcccSDK;
}
// 设置TCCC事件回调
[self.tcccSDK addTcccListener:self];
// 移除TCCC事件回调
[self.tcccSDK removeTCCCListener:self];
// 销毁实例
[TCCCWorkstation destroySharedIntance];
__tcccSDK = nil;
```

#### C++

```
finclude "TCCCSDE/tccc/include/TTCCCWorkstation.h"
using namespace tccc;
// 创建实例和设置事件回调
TTCCCWorkstation* tcccSDE = getTCCCShareInstance();
// 设置回调, TCCCCallbackImpl 需要继承 TTCCCCallback
class TCCCCallbackImpl mage#承 TTCCCCallback {
    public:
        TCCCCallbackImpl() {}
        -TCCCCallbackImpl() {}
        void onError(TCCCError errCode, const char* errMag, void* extraInfo) {}
        void onError(TCCCCWarning warningCode, const char* warningMsg, void*
extraInfo) {}
        void onEnded(EndedReason reason, const char* reasonMessage, const char*
sessionId) {}
    };
    TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();
    tcccSDK->addCallback(tccCallback);
// 销数实例
destrorTCCCSDstreInstance();
tcccSDK = nullptr;
```

#### 登录相关接口

API

描述



login	SDK 登录。
checkLogin	检查 SDK 是否已登录。
logout	SDK 退出登录。

#### 登录、退出登录示例代码

Swift

```
// 登录的坐席ID,通常为邮箱地址
// 腾讯云联络中心应用ID,通常为1400开头
  // 登录成功
// 检查登录状态
```

Objective-C

// **引入** oc 头文件



```
TXLoginParams *param = [[TXLoginParams alloc] init];
// 登录的坐席ID, 通常为邮箱地址
param.userId = @"";
// 登录票据, 在登录模式为Agent必填。更多详情请参见(创建 SDK 登录
// Token](https://cloud.tencent.com/document/product/679/49227)
param.token = @"";
// 腾讯云联络中心应用ID, 通常为1400开头
param.sdkAppId = 0;
// 设置为坐席模式
param.type = Agent;
[self.tcccSDK login:param succ:^(TXLoginInfo * _Nonnull info) {
    // 登录成功
} fail:^(int code, NSString * _Nonnull desc) {
    // 登录失败
}];
// 检查登录状态
[self.tcccSDK checkLogin:^{
    // 已登录
} fail:^(int code, NSString * _Nonnull desc) {
    // 未登录或者被T7
}];
// 遇出磁功
fail:^(int code, NSString * _Nonnull desc) {
    // 退出成功
} fail:^(int code, NSString * _Nonnull desc) {
    // 退出展常
];
```

#### C++

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
// 登录回调类
class TCCCLoginCallbackImpl : public ITXValueCallback<TCCCLoginInfo> {
public:
    TCCCLoginCallbackImpl() {
        }
        ~TCCCLoginCallbackImpl() override {}
        void OnSuccess(const TCCCLoginInfo &value) override {
            // 登录成功
        }
        void OnError(TCCCError error_code, const char *error_message) override
```



// 登录失败
}
};
TCCCLoginCallbackImpl* loginCallBackImpl = nullptr;
if (nullptr == loginCallBackImpl) {
<pre>loginCallBackImpl = new TCCCLoginCallbackImpl();</pre>
}
TCCCLoginParams param;
/// <b>登录的坐席</b> ID <b>,通常为邮箱地址</b>
<pre>param.userId = "";</pre>
/// <b>登录票据,在登录模式为</b> Agent <b>必填。更多详情请参见[创建</b> SDK <b>登录</b>
/// Token](https://cloud.tencent.com/document/product/679/49227)
<pre>param.token = "";</pre>
/// <b>腾讯云联络中心应用</b> ID <b>,通常为</b> 1400 <b>开头</b>
<pre>param.sdkAppId = 0;</pre>
// <b>设置为坐席模式</b>
<pre>param.type = TCCCLoginType::Agent;</pre>
// <b>登录</b>
<pre>tcccSDK-&gt;login(param,loginCallBackImpl);</pre>
(1)// <b>退出登录</b>
tcccSDK->logout (nullptr);

# 呼叫相关接口函数

API	描述
call	发起通话。
answer	接听来电。
terminate	结束通话。
sendDTMF	发送 DTMF(双音多频信号 )。
mute	静音。
unmute	取消静音。

#### 发起呼叫和结束呼叫示例代码

Swift import TCCCSDK let callParams = TXStartCallParams() // 呼叫的手机号 callParams.to = ""; // 号码备注,在通话条中会替代号码显示(可选 callParams.remark = "";



# // 发起外呼 tcccSDK.call(callParams) { // 发起呼叫成功 } fail: { code, message in // 发起呼叫失败 } // 结束通话 tcccSDK terminate()

#### Objective-C

```
// 引入 oc 头文件
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

TXStartCallParams *callParams = [[TXStartCallParams alloc] init];
// 呼叫的手机号
callParams.to = TO;
// 号码备注,在通话条中会替代号码显示(可选)
callParams.remark = @"testByIos";
// 发起外呼
[self.tcccSDK call:callParams succ:^{
    // 发起呼叫成功
} fail:^(int code, NSString * _Nonnull desc) {
    // 发起呼叫失败
}];

// 结束通话
[self.tcccSDK terminate];
```

#### C++

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
class TCCCCommonCallback : public ITXCallback {
private:
    NSString* mFunName;
public:
    TCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    ~TCCCCommonCallback() override {
    }
    yoid OnSuccess() override {
}
```





#### 音频设备接口函数

API	描述
setAudioCaptureVolume	设定本地音频的采集音量。
getAudioCaptureVolume	获取本地音频的采集音量。
setAudioPlayoutVolume	设定远端音频的播放音量。
getAudioPlayoutVolume	获取远端音频的播放音量。
setAudioRoute	设置音频路由。

# 切换音频路由示例代码

#### Swift

```
import TCCCSDK
// 切换为扬声器
tcccSDK.getDeviceManager().setAudioRoute(.TCCCAudioRouteSpeakerphone)
// 静音
tcccSDK.mute()
// 取消静音
tcccSDK.unmute()
```



#### Objective-C

# // 引入 oc 头文件 #import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h" // 切换为扬声器 [[self.tcccSDK getDeviceManager] setAudioRoute:TCCCAudioRouteS // 静音 [self.tcccSDK mute]; // 取消静音 [self.tcccSDK unmute];

#### C++

```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
// 切換为扬声器
tcccSDK->getDeviceManager()-
>setAudioRoute(TCCCAudioRoute::TCCCAudioRouteSpeakerphone);
// 静音
tcccSDK->mute();
// 取消静音
tcccSDK->unmute();
```

# 调试相关接口

API	描述
getSDKVersion	获取 SDK 版本信息。
setLogLevel	设置 Log 输出级别。
setConsoleEnabled	启用/禁用控制台日志打印。
callExperimentalAPI	调用实验性接口。

#### 获取SDK版本示例代码

Swift

import TCCCSDK



#### // **获取**SDK**版本号**

et version = TCCCWorkstation.getSDKVersion()

#### Objective-C

// 51へ OC 天文件 #import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"

// 获取SDK版本号
NSString\* version = [TCCCWorkstation getSDKVersion];

#### C++

#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;

// 获取SDK 版本号

#### 错误和警告事件

API	描述
onError	错误事件回调。
onWarning	警告事件回调。

#### 处理错误回调事件回调示例代码

Swift	
import TCCCSDV	
<pre>func onError(_ errCode: TCCCErrorCode, errMsg: String, extInfo: [AnyHashable :</pre>	
Any]?) { // <b>错误事件回调</b>	
}	
[AnyHashable : Any]?) { // 警告事件回调	
[AnyHashable : Any]?) { // <b>警告事件回调</b> }	



#### · 设置TCCC事件回调

tcccSDK.addTcccListener(self)

#### Objective-C

```
// 引入 oc 头文件
#import "TCCCSDK/tccc/platform/apple/TCCCWorkstation.h"
#pragma mark - TCCCDelegate
- (void)onError:(TCCCErrorCode)errCode errMsg:(NSString * _Nonnull)errMsg extInfo:
(nullable NSDictionary *)extInfo {
    // 错误事件回调
}
- (void)onWarning:(TCCCCWarningCode)warningCode warningMsg:(NSString
*_Nonnull)warningMsg extInfo:(nullable NSDictionary *)extInfo {
    // 警告事件回调
}
// 设置TCCC事件回调
[self.tcccSDK addTcccListener:self];
```

#### C++



#### 呼叫相关事件回调


API	描述
onNewSession	新会话事件。包括呼入和呼出。
onEnded	会话结束事件。
onAudioVolume	音量大小的反馈回调。
onNetworkQuality	网络质量的实时统计回调。

#### 处理接听和坐席挂断事件回调示例代码

Swift

```
import TCCCSDK
func onNewSession(_ info: TXSessionInfo) {
    // 新会话事件。包括呼入和呼出
}
func onAccepted(_ sessionId: String) {
    // 对端已接听事件
}
func onEnded(_ reason: TXEndedReason, reasonMessage: String, sessionId: String) {
    // 通话结束事件
}
// 设置Tccc事件回调
tcccSDK.addTcccListener(self)
```

#### Objective-C





#### self.tcccSDK addTcccListener:self];

#### C++

# 与云端连接情况的事件回调

API	描述
onConnectionLost	SDK 与云端的连接已经断开。
onTryToReconnect	SDK 正在尝试重新连接到云端。
onConnectionRecovery	SDK 与云端的连接已经恢复。

#### 与云端连接情况的事件回调示例代码

Swift	
import TCCCSDK	
func onConnectionLost(_ serverType: TXServerType) {     // SDK <b>与云端的连接已经断开</b> }	
func onConnectionRecovery(_ serverType: TXServerType) {	
func onTry(toReconnect serverType: TXServerType) {	



# // SDK 正在尝试重新连接到云端

// 设置TCCC事件回调

tcccSDK.addTcccListener(self)

#### Objective-C



#### C++





# API 错误码

# 基础错误码

符号	值	含义
ERR_SIP_SUCCESS	200	成功。
ERR_UNRIGIST_FAILURE	20001	登录失败。
ERR_ANSWER_FAILURE	20002	接听失败,通常是 trtc 进房失败。
ERR_SIPURI_WRONGFORMAT	20003	URI 格式错误。
ERR_HTTP_REQUEST_FAILURE	-10001	Http 请求失败,请检查网络连接情况。
ERR_HTTP_TOKEN_ERROR	-10002	token 登录票据不正确或者已过期。
ERR_HTTP_GETSIPINFO_ERROR	-10003	获取坐席配置失败,请 <mark>联系我们</mark> 。

# SIP 相关错误码

符号	值	含义
ERR_SIP_BAD_REQUEST	400	错误请求。
ERR_SIP_UNAUTHORIZED	401	未授权(用户名密码不对情况)。
ERR_SIP_PAYMENTREQUIRED	402	许可不足,需要增加或者购买坐席许可。
ERR_SIP_AUTHENTICATION_RE QUIRED	407	代理需要认证,请检查是否已经调用登录接口。
ERR_SIP_REQUESTTIMEOUT	408	请求超时(网络超时)。
ERR_SIP_REQUEST_TERMINATE D	487	请求终止(网络异常,网络中断场景下)。
ERR_SIP_SERVICE_UNAVAILAB LE	503	服务不可用。
ERR_SIP_SERVER_TIMEOUT	504	服务超时。

# 音频设备相关错误码

符号	值	含义
ERR_MIC_START_FAIL	-1302	打开麦克风失败。设备,麦克风的配置程序(驱动程序)异 常,禁用后重新启用设备,或者重启机器,或者更新配置程 序。
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权,通常在移动设备出现,可能是权限被用户 拒绝了。
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败。



ERR_MIC_OCCUPY	-1319	麦克风正在被占用中,例如移动设备正在通话时,打开麦克风 会失败。
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败。
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败,例如在 Windows 或 Mac。
ERR_SPEAKER_SET_PARAM_FA IL	-1322	扬声器设置参数失败。
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败。
ERR_UNSUPPORTED_SAMPLER ATE	-1306	不支持的音频采样率。

# 网络相关错误码

符号	值	含义
ERR_RTC_ENTER_ROOM_FAILE D	-3301	进入房间失败,请查看 onError 中的 −3301 对应的 msg 提 示确认失败原因。
ERR_RTC_REQUEST_IP_TIMEO UT	-3307	请求 IP 和 sig 超时,请检查网络是否正常,或网络防火墙是否 放行 UDP。
ERR_RTC_CONNECT_SERVER_ TIMEOUT	-3308	请求进房超时,请检查是否断网或者是否开启 vpn,您也可以 切换 4G 进行测试确认。
ERR_RTC_ENTER_ROOM_REFU SED	-3340	进房请求被拒绝,请检查是否连续调用 enterRoom 进入相同 ID 的房间。



# 实现一键外呼 Web

最近更新时间: 2024-12-02 10:01:22

## 步骤1:初始化 SDK

初始化方法详情请参见 初始化SDK。

#### ▲ 注意:

后续步骤需要在 tccc.events.ready 事件成功后才能执行。

# 步骤2:实现点击按钮触发SDK外呼

# 

#### React

```
import { useState } from 'react';
export function CallButton() {
  const [phoneNumber, setPhoneNumber] = useState('1999999999') // 请替换为真实外呼号码
  function sdkCall(phoneNumber) {
    window.tccc.Call.startOutboundCall({
      phoneNumber,
    }).then((res) => {
```



```
}).catch((err) => {
    const error = err.errorMsg;
    })
}
return (
    <button onClick={sdkCall}>一键外呼</button>
)
}
```

#### 原生JS

```
<button id="call">- ### /button>
<script>
function sdkCall(phoneNumber) {
    window.tccc.Call.startOutboundCall({
        phoneNumber, // 外呼号码
        phoneDesc: 'Tencent' // 备注文案, 将会在通话条上替代号码的显示
    }).then((res) => {
        // 外呼成功,并获取外呼ID,后续可用作查询关联录音、服务记录信息
        const sessionId = res.data.sessionId
    }).catch((err) => {
        // 外呼失败,获取失败原因并提示
        console.error(err.errMsg)
     })
    // 监听按钮的点击事件,并触发外呼方法
    document.getElementById('call').addEventListsner('click', () => {
        // 请替换为真实外呼号码
        sdkCall('1999999999');
    })
```

成功触发外呼后,等待对方接听,依次触发相关事件。

# 外呼事件流程









# 小程序

最近更新时间: 2025-06-23 15:02:52

# 介绍

腾讯云联络中心提供外呼小程序,您可以以下几种方式来打开小程序进行外呼电话:

- 小程序内 打开小程序 进行外呼。
- 企业微信 跳转小程序 进行外呼。
- 自有 App 拉起小程序 进行外呼。

# 效果展示



# 参数说明

# 公共参数说明

拉起外呼小程序需要用到以下参数。 请先记录这些参数,后文将会详细介绍具体开发方式。

参数	值	备注
appld	wx49e8ab828096cff3	固定值。
path	/pages/call/call? sdkAppId=140000000&userId=Foo	请将具体参数值修改为您自己的 path。



	OrBar@tencent.com&calleePhone=1 99999999999&calleeRemark=Display Name&uui=HelloWorld	
id	gh_bafb4b7e104a	腾讯云联络中心小程序原始 ID,仅 App 拉起小程序 时使用。

# path 参数说明

参数	类型	必填	备注
sdkAppId	String	是	腾讯云联络中心应用 SDKAppID。
userld	String	是	座席账号,一般为邮箱格式。
calleePhone	String	是	需要呼叫的电话号码。
calleeRemark	String	否	该电话的备注文案,可替代号码显示。
uui	String	否	自定义参数,传入后将通过 CDR 事件推送 返回。
token	String	否	使用 Token 方式 快速登录外呼小程序,建议每次发起呼叫的时候都重新从 TCCC 获取。
skillGroupId	String	否	指定技能组内绑定的外呼号码。
callerPhoneNum ber	String	否	指定外呼号码。
servingNumberGr oupIds	String[ ]	否	指定号码 ID 列表。
innerUserId	String	否	指呼叫内线的员工邮箱或 SIP 话机,当 calleePhone 参数为 null 时,该 参数才生效。 • 建议同时使用 calleeRemark 显示对方姓名。 • 员工邮箱格式为: xxx@xx.com , SIP 话机格式为: 1011@1400xxx.tccc.qcloud.com 。
phoneEncodeTyp e	'numb er'	否	目前仅支持'number',在开启 <mark>号码映射</mark> 时强制使用真实号码。
soundMode	String	否	指定声音模式,默认为扬声器。 • speaker:表示扬声器。 • ear:表示听筒模式。
autoExitTime	'numb er'	否	指定对端挂断后多少秒后自动退出TCCC小程序,值范围[-1, 30]。 • 默认值为 -1,即表示不主动退出。 • 若值为 0 即立刻退出。 • 若值为2.5的时候即2.5秒后自动退出。

# 接入方式

# 小程序内打开小程序进行外呼

推荐使用 wx.navigateToMiniProgram 跳转打开小程序,具体接入指引请参见 跳转小程序



#### 可直接点击 外呼小程序代码片段 使用,使用方式请参见 导入代码片段

代码示例:

#### 企业微信里进行外呼

#### 情景一:企业微信应用是网页

```
() 说明:
```

前提条件:开发前需要联系官方人员授权小程序:点此进入 TCCC 社群联系官方人员,否则打开小程序会报错 "not allow to cross corp"。

#### 操作步骤:

1. 通过 wx.agentConfig 注入应用的权限。

#### 🕛 说明:

- 如果通过新版的 wecom-jssdk.js 来实现注入应用的权限,需要再调用 ww.register 的时候指定 getAgentConfigSignature,非 getConfigSignature。
- 如果用 getConfigSignature 的话, 会报 "launch Miniprogram:fail\_no permission to execute".
- 3. 调用 launchMiniprogram 打开小程序。
   代码示例:

```
const sdkAppId = '140000000'
const userId = 'userid@email.com'
const calleePhone = '19999999999'
wx.invoke('launchMiniprogram', {
    appid : "wx49e8ab828096cff3",
    path : `pages/call/call?
sdkAppId=${sdkAppId}&userId=${userId}&calleePhone=${calleePhone}`,
}, function(res) {
    if(res.err_msg == "launchMiniprogram:ok") {
```





#### 情景二:企业微信应用是小程序

推荐使用 wx.navigateToMiniProgram 跳转打开小程序。 代码示例:

# App 内打开小程序外呼

使用微信提供的 OpenSDK 拉起外呼小程序,请参见 App 拉起小程序功能 。

- Android 开发示例
- iOS 开发示例



# uni-app

最近更新时间: 2023-06-19 10:23:02

## 呼叫相关接口函数

API	描述
call	发起通话
answer	接听来电
terminate	结束通话
sendDTMF	发送 DTMF(双音多频信号)
mute	静音
unmute	取消静音
startPlayMusic	开始播放音乐
stopPlayMusic	停止播放音乐

#### 发起呼叫和结束呼叫示例代码

```
// 发起呼叫,发起呼叫前请先调用登录接口。tcccSDK.login
tcccSDK.call({
   to: '134xxxx', // 被叫号码(必填)
   remark: "xxx", // 号码备注,在通话条中会替代号码显示(可选)
   uui: "xxxx", // 户自定义数据(可选)
}, (code,message) => {
   if (code == TcccErrorCode.ERR_NONE) {
      // 发起成功
   } else {
      // 发起失败
   }
});
// 结束通话
tcccSDK.terminate();
// 接听来听
tcccSDK.answer((code,message) => {
   if (code == TcccErrorCode.ERR_NONE) {
      // 接听成功
   } else {
      // 接听失败
   }
});
```



# Android

最近更新时间: 2025-04-22 14:23:22

# 呼叫相关接口函数

API	描述
call	发起通话
answer	接听来电
terminate	结束通话
sendDTMF	发送 DTMF(双音多频信号)
mute	静音
unmute	取消静音

# 发起呼叫和结束呼叫示例代码

```
TCCCTypeDef.TCCCStartCallParams callParams =new TCCCTypeDef.TCCCStartCallParams();
//其中1343xxxx为手机号
callParams.to = "13430xxxx";
// 发起通话,发起呼叫前请先调用登录接口。tcccSDK.login
tcccSDK.call(callParams, new TXCallback() {
    @Override
    public void onSuccess() {
        // call success
     }
    @Override
    public void onError(int code, String desc) {
        // call error
     }
});
// 结束通话
tcccSDK.terminate("");
```

# 其他

腾讯云联络中心还推出的一款电话外呼、接听的云联络中心 UI 组件,通过集成该组件,您只需要编写几行代码就可以为您的 App 添加外 呼手机、电话等音频通话等功能。目前 TCCCCallKit 支持 Android 、iOS 等多个开发平台。

- 腾讯云联络中心含 UI 解决方案(iOS)
- 腾讯云联络中心含 UI 解决方案 (Android)



# iOS

最近更新时间: 2025-04-22 14:23:22

# 呼叫相关接口函数

API	描述
call	发起通话
answer	接听来电
terminate	结束通话
sendDTMF	发送 DTMF(双音多频信号)
mute	静音
unmute	取消静音

# 发起呼叫和结束呼叫示例代码

```
class TCCCCommonCallback : public ITXCallback {
private:
    NSString* mFunName;
public:
    TCCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    -TCCCCcommonCallback() override {
     }
    void OnSuccess() override {
     // 成功
    }
    void OnError(TCCCError error_code, const char *error_message) override {
        std::string copyErrMsg = makeString(error_message);
        // 失败
    }
};
TCCCCommonCallback* startCallCallbackImpl = nullptr;
if (nullptr == startCallCallbackImpl) {
    startCallCallbackImpl = new TCCCCommonCallback(@"startCall");
}
TCCCStartCallParams callParams;
//FFM的手机号
callParams.to = "";
// 发起外呼,发起呼叫前请先调用登录接口。tcccSDK->login
tcccSDK->call(callParams, startCallCallbackImpl);
// 结束通话
```



#### tcccSDK->terminate();

# 其他

腾讯云联络中心还推出的一款电话外呼、接听的云联络中心 UI 组件,通过集成该组件,您只需要编写几行代码就可以为您的 App 添加外 呼手机、电话等音频通话等功能。目前 TCCCCallKit 支持 Android 、iOS 等多个开发平台。

- 腾讯云联络中心含 UI 解决方案(iOS)
- 腾讯云联络中心含 UI 解决方案(Android)



# 实现电话呼入 Web

最近更新时间: 2023-04-24 09:41:00

# 初始化 SDK

请参见 初始化 SDK。

#### ▲ 注意:

后续步骤需要在 tccc.events.ready 事件成功后才能执行。

# 接听方式

#### 方式1: SDK API 接听

1. 通过 tccc.on 绑定电话呼入事件 tccc.events.callIn 来监听电话呼入,获取sessionId;

```
2. 使用 tccc.Call.accept() 来主动接听。
```

#### 参考示例代码:

```
let sessionId; //存在公共区域,可以方便任意时候使用
// 监听电话呼入事件
window.tccc.on(window.tccc.events.callIn, (response) =>
 // 会话呼入时触发,将该会话的sessionId存储到公共区域
    sessionId = response.data.sessionId;
})
// 实现接听方法
function accept() {
    if (sessionId) {
        window.tccc.Call.accept({ sessionId })
        .then(() => {
            // 接听成功,开始通话
            })
        .catch(err => {
            // 接听失败,展示详细错误原因
            const error = err.errorMsg;
            })
        } else {
            console.error('未找到需接听的会话');
        }
        // 之后,可以在需要的地方执行 accept() 来触发接听电话
```

## 方式2:点击通话条接听



۲	Q					# 🔹 🇐
<b>TCCC Demo</b> 当前环境:正式环境	客户列表		点击接听按钮	请求电话接入中 Gahouliao 广东馬州移动		-
■計概览 ※ 客户列表	Q、搜索客户			L 接	uff (4)	
▲ 产品列表 ▲ 账号	□ 姓名	邮箱	地址	号码	注册日期	操作
✿ 设置	• • •	CENTRAL IN	武京市	038546***83	2021/12/24	一键外呼
	□ () *7	RT INDUSTRIAL	海南市	0367025***80	2022/11/07	一键外呼
	0 0 **	11-87.800,011-10	海南市	1914***5982	2022/10/10	一键外呼
	• • •	1210-04	诸码市	0279096***86	2022/02/28	一键外口
		1-200 (Stational con-	贵宁市	1953***4390	2022/12/05	一键外呼

# 其他相关事件

```
window.tccc.on(window.tccc.events.callIn, (response) => {
    // 会话呼入时触发
})
window.tccc.on(window.tccc.events.userAccessed, (response) => {
    // 座席接入
})
window.tccc.on(window.tccc.events.sessionEnded, (response) => {
    // 会话结束时触发
})
```



# 小程序/Android/iOS

最近更新时间: 2023-04-24 09:41:00

在所有移动端,都建议座席使用"手机接听"的方式,这样接听电话更加及时。可以由管理员在工作台开启手机接听配置,具体流程: 1. 登录 TCCC管理端,依次进入 电话客服−>号码管理−>白名单审批,新建申请单(可以参考文档 首次登录管理端 进行登录)。



2. 填写白名单申请单,之后等待审批通过。



3. 将需要的号码开启手机接听,依次进入 号码管理 > 编辑,在最底部开启"可用于座席手机接听"。



📞 电话外呼 📔 🗌	лға 🕂 бина 🕓 дууда	🖬 新手配置指引	<b>2</b> 2
■ 概览	号码管理 如您在使用过程中遇到任何问题,欢迎随时联系我们,咨询热线:0755-38564058,或微信扫码加入TCCC技术服务交流群。		
☺ 数据分析	号码管理 白名单审批 白名单查看		
➡ 服务记录	2. 进入号码管理		
④ 实时监控	全部 (4 未标记号码 14) ttim (0) 能用的号码 (0) +新建号码标签		
○ 在线客服			
▲ 电话客服 ~	起量编辑		Q
<ul> <li>技能组管理</li> </ul>	电话号码 呼叫类型 号田状态 归属地 号田标签 号田心 关联的 外听技 工作时间 满意	度 号码并 操作	F
- IVR管理	16 J MP	3. 点击结	扁辑
、号码管理 1. 进入电		度:不限 武満意 制	>
· 电话客服设置	0098602062963618 以冲出 12/用中 广东厂 1049 評出WR: 度设 adrien	Ptfr:- 每秒号码 并发度:	/
<ul> <li>自助服务</li> </ul>	test/1.1	不限制	
• 放音文件管理	野入IVR: 耳色小me	号码并发 度:不限	
<ul> <li>智能质检</li> </ul>	T7KPim     F7KPim     F7KP	出满意 制 所介:- 毎秒号码 編編	
• 话机与内线号码管理	飲い	开发度: 不限制	

#### 开启手机选项:

	请输入呼入失败放首
	▶ 点击试听 ▲ 在最底部开启该选项
通用设置	
可用于座席手机通话	开启后, 座席使用"手"接听"与"手机呼出"功能时,将优先使用该号码呼叫座席手机。

#### 4. 通知座席在工作台开启手机接听:



至此,您已经完成了手机端接听的配置。

# 常见问题 小程序端常见问题

最近更新时间: 2025-05-20 17:42:02

# 跳转到 TCCC 小程序发起外呼的形式外,TCCC 还会提供其他方式吗?

因通话需要用到微信的 live-pusher 组件,该组件权限需要申请开通对应的类目,如果您的企业可以申请下来并且需要 sdk 或者插件 的方式,您可以 联系我们 。

#### 可以去掉跳转提醒吗?

```
如下图所示,该跳转提醒可去掉吗?
```



答:在微信下无法去掉,但在企业微信下可以去掉。

实现思路是:把 **腾讯云联络中心** 小程序添加到企微工作台,然后把 **腾讯云联络中心** 小程序的可见范围设置到普通用户不可见。具体步骤 如下:

- 企业微信管理员登录 企业微信管理后台,单击创建应用 > 已有小程序快速创建,随后弹窗会出现一个二维码,您需 点此进入 TCCC 社群后联系腾讯云联络中心的小程序管理员扫码即可添加。
- 2. 最后把该应用的可见范围设置为一个空部门或者是仅某人。这样普通用户就不会有该调整提醒了。

# 通话过程中可以切后台或退出 TCCC小程序吗?

可以。点击位于通话页面的右上角按钮即可(如下图所示按钮),点击后通话将正常进行并且不中断。



#### 在企业微信中退出 TCCC 后,通话依然在进行中,但无悬浮球,怎么解决?

答:因企业微信暂时还未支持悬浮球,但可通过额外的开发来方式来实现下面类似的悬浮球功能,请<mark>联系我们</mark>获取技术方案。





## 可在 PC 上使用吗?

可以,目前已有客户在 Windows、macOS、Android、iOS 系统使用。

#### 提示网络很差或者异常应该怎么解决?



通常情况下是本地网络有短暂的异常,建议您尝试打开飞行模式,然后重新开启后重试。如使用了代理软件请关闭代理软件。

#### 声音音质差或者声音忽大忽小?

- 1. 首先检查一下本地网络有没有问题,可尝试打开飞行模式,然后再重新开启。
- 2. 检查一下是否有其他 App 开启是音视频播放的软件未关闭。关闭类似的 App。
- 3. 也可能是对端的手机信号不好引发,可呼叫朋友或同事的手机号进行测试,如持续未解决可联系我们。

#### 提示未开启麦克风权限异常,怎么样处理?

如下图所示:



客户端错误
未开启麦克风权限。请检查麦克风 是否被占用,以及麦克风权限是否 开启后重试
挂断

答:通常情况下,用户同意申请麦克风权限后不会有该问题,仅仅是用户拒绝或手动禁用麦克风后才需要手动开启麦克风权限。 请按如下步骤操作:

- 1. 进入手机设置:打开手机「设置」>向下滑动找到「应用管理」(部分机型为"应用设置"或"应用与权限"),点击进入应用列表
- 2. 定位企微或者微信应用:在列表中找到"企微或微信">点击进入「应用信息」页面>选择「权限管理」(或"权限"选项)
- 3. 开启麦克风权限:在权限列表中找到「麦克风」>点击右侧开关,将其切换为"允许"状态(绿色为开启)
- 4. 确认生效后重新尝试通话功能。

#### 提示登录票据过期,应该怎么解决?

报如下登录票据过期错误:

登录失败	
登录票据过期,请重新登录	
确认	

- 1. 如果使用 token 登录的话, token 的有效期是 10 分钟,超过 10 分钟后使用就会报该错误。这里建议每次发起跳转的时候都从后台 获取登录 token 重新,业务侧不需要缓存该 token。
- 2. 如果是使用微信快捷或者账号密码登录的话,登录超过了 24 小时没有操作就会报该错,重新登录即可。

#### 外呼对端挂断后是否可自动返回业务小程序?

可以,在外呼的时候指定参数 autoExitTime。该参数可实现对端挂断后多少秒后自动退出 TCCC 小程序并返回业务小程序。

# Web SDK 常见问题

最近更新时间: 2023-07-25 15:27:52

## TCCC Web SDK 支持什么框架?

TCCC Web SDK 是纯 JavaScript 实现的,支持运行在 Vue、React、uni-app、PHP、JSP 等环境。

#### SDK 的界面支持展示其他信息吗?

不支持。

#### SDK 的通话条按钮可以隐藏吗?

支持。

## 初始化 SDK 时,Userld 是什么?

Userld 就是腾讯云联络中心里面的账号,一般为邮箱格式,可以在控制台或者管理后台创建。

#### SDK 怎么切换账号?

重新使用不同的Userld初始化SDK,会自动切换账号。

#### 为什么使用 SDK 要使用 HTTPS 部署页面?

因为浏览器的限制,只能在 HTTPS 下获取麦克风权限。

#### 外呼时如何指定外显号码?

界面上不支持指定,在调用 SDK 外呼 API 时可指定外显号码。

#### Token 需要续期吗,过期了怎么办?

SDK 初始化完成后,不需要续期 Token,请开发者确保初始化 SDK 时保证 Token 在有效期内。

#### 登录之后提示设备错误

- 1. 检查网站 URL 是否为 HTTPS。
- 2. 检查是否允许麦克风权限。
- 3. 使用 检测网站,按照步骤执行。
- 4. 开发可以根据 SDK 提供的 API, isBrowserSupported 和 isEnvSupported 做自定义提示。





## 呼入时无响铃

如果呼入时,SDK 的页面最小化或者切换到其它页面,由于 浏览器的限制,可能会出现无响铃的现象,建议开启浏览器通知,或者通过 监听 SDK callIn 事件,业务侧做一个强提示处理。

## 外呼失败

SDK 初始化之后,需要等待 ready 事件之后才能外呼。另外请确保实例的号码列表中有能够外呼的号码。

#### 通话中突然中断

根据 SDK sessionEnded 事件的 closeBy 字段判断是哪一方挂断。



# uni-app SDK 常见问题

最近更新时间: 2023-12-15 15:34:41

# 如何查看 TCCC 日志?

TCCC 的日志默认压缩加密,后缀为 .log。

日志路径:

- Android: /sdcard/Android/data/包名/files/tccc
- iOS:在 sandbox/Documents 目录下的 tccc 文件夹

# TCCC SDK 在 Android 能不能支持X86模拟器?

TCCC 目前版本暂时不支持,未来会支持模拟器。如果需要在模拟器运行,建议在 iOS 下的 x86 模拟器上运行调试。

## TCCC SDK 在 iOS 能不能支持 armv7 的 CPU 类型?

因在 iPhone 5c 以下才有该类型的 CPU,目前基本上已经无人使用了。所以我们不适配该类型的 CPU,并且在云打包 iOS 的时候需 要修改配置 manifest.json 文件。

```
"validArchitectures": [
"arm64"
],
```

# 为什么 iOS 下手机切后台通话中断?

因手机应用程序在切换到后台时,操作系统会暂停应用程序的进程以节省资源。可以在 iOS 下需要配置 audio background mode 才 可以保证有音频影响的时候程序不会终止。



# 为什么手机下能不能处理呼入?



如果手机在前台运行的时候有新会话将会收到 onNewSession 回调,但是我们不建议您在手机上处理呼入(App 在切换到后台时会暂 停程序),建议您开通手机接听功能。



# 客户端 SDK 常见问题

最近更新时间: 2025-05-20 17:42:02

## 如何查看 TCCC 日志?

TCCC 的日志默认压缩加密,后缀为 .log。

- Android 日志路径: /sdcard/Android/data/包名/files/tccc
- iOS 日志路径: sandbox/Documents/tccc

#### TCCC Agent Android 端能不能支持模拟器?

TCCC 目前版本暂时不支持,未来会支持模拟器。

#### Android 退后台停止音频采集

Andorid 9.0系统对 App 退后台的麦克风做了限制,为防止通话的时候程序退后台引起的通话被静音问题。请在 App 退后台情况下发 送前台通知来防止通话被静音,或参考我们开源的 TCCCCallKit 方案来解决。

#### 如何在项目中同时使用 TCCC 和 TRTC?

如果在项目中同时使用了 TCCC 和 TRTC 编译会报错,因为是 TCCC 也是引入了部分 TRTC 的库,所以需要在编译的时候把共用的 部分排除 。

#### Android

1. 去掉相同的 jar 包,重新生成 tcccswap-release-trtc-fixed.aar 包。



#### () 说明:

编译的时候报 zipxception: only DEFLATED entries can have EXT descriptor 错误,大概由于压缩格式不兼 容导致(使用了第三方的压缩软件)。请用上面的命令行来压缩,或者使用 Android Studio 自带的 ZIP 工具。

2. 修改引入的 TRTC 的版本,保证 tccc 和 trtc 同一个版本(如果版本不一致会引发 crash)。如果不确定版本号,请 联系我 们 。

implementation 'com.tencent.liteav:LiteAVSDK\_TRTC:12.1.0.14886'

3. 排除 trtc 和 tccc 共用的 so 文件以保证编译通过。 修改 build.gradle 文件,把下面代码添加到 android 节点下:

pa	ckaging0pt	ions {
	pickFirst	"lib/arm64-v8a/libliteavsdk.so"
	pickFirst	"lib/arm64-v8a/libtxffmpeg.so"
	pickFirst	"lib/arm64-v8a/libtxsoundtouch.so"
	pickFirst	"lib/armeabi-v7a/libliteavsdk.so"
	pickFirst	"lib/armeabi-v7a/libtxffmpeg.so"
	pickFirst	"lib/armeabi-v7a/libtxsoundtouch.so"

#### iOS

建议手动引入 TCCC SDK,请不要用 CocoaPods 的方式引入,具体操作方法请参见 集成 SDK - 方案2:手动下载。也请不要添加 txsoundtouch.xcframework 和 txffmpeg.xcframework。

#### 在 iOS 下回调是否都在主线程

Swift、OC 接口的所有回调均在主线程,开发者无需特别处理。但 C++ 接口下回调都不在主线程,需要业务层面上判断并且把他转为主 线线程:

```
if ([NSThread isMainThread]) {
    // 在主线程,直接可以处理
    return;
}
dispatch_async(dispatch_get_main_queue(), ^{
    // 回调在非主线程。
});
```

#### 为什么手机下能不能处理呼入?

如果手机在前台运行的时候有新会话将会收到 onNewSession 回调,但是我们不建议您在手机上处理呼入(App 在切换到后台时会冻 结程序),建议您开通手机接听功能。

#### 其他平台如 Windows 有没有对应的 SDK?

TCCC 提供了全平台 SDK,如有需要可 联系我们,我们线下提供。

# 集成在线会话用户端 SDK Web(vue2/vue3)

最近更新时间: 2025-07-11 15:02:31

# 介绍

腾讯云

智能客服用户端 Web UlKit。使用此 UlKit,您可以在一天内将智能客服的能力集成到您的 Web 或 Hybrid 项目。极简接入,用智能 客服为您的产品增收提效,提升用户满意度和付费转化。

# 效果展示



# 开发环境要求

- Vue (全面支持 Vue2 & Vue3 ,请您在下方接入时选择您所匹配的 Vue 版本接入指引进行接入)
- TypeScript(如果您是 js 项目, 请参见 常见问题- js 工程如何接入 TUIKit 组件 进行配置 ts 渐进式支持 )
- sass (sass-loader 版本 ≤ 10.1.1)
- node ( node.js ≥ 16.0.0 )
- npm(版本请与 node 版本匹配)

# UIKit 源码集成

#### 步骤1: 创建项目

支持使用 webpack 或 vite 创建项目工程,配置 Vue3 / Vue2 + TypeScript + sass。以下是几种项目工程搭建示例:

vue-cli



🕛 说明:

1. 请确保您的 @vue/cli 版本在 5.0.0 以上,您可使用以下示例代码升级 @vue/cli 版本至 v5.0.8。

2. 如果您的项目由较低版本的 @vue/cli 创建,集成 UIKit 后运行项目如有报错,请查阅 常见问题 解决。

使用 vue-cli 方式创建项目, 配置 Vue2 / Vue3 + TypeScript + sass。 如果您尚未安装 vue-cli 或者 vue-cli 版本低于 5.0.0, 可以在 terminal 或 cmd 中采用如下方式进行安装:

m install -g @vue/cli@5.0.8 sass sass-loader@10.1.1

通过 vue-cli 创建项目,并选择下图中所选配置项。

vue create ai-desk-example

请务必保证按照如下配置选择:



创建完成后,切换到项目所在目录:

cd ai-desk-example

如果您是 vue2 项目,请根据您所使用的 Vue 版本进行以下相应的环境配置, vue3 项目请忽略。

vue2.7

npm i vue@2.7.9 vue-template-compiler@2.7.9

vue2.6及以下

腾讯云

npm i @vue/composition-api unplugin-vue2-script-setup vue@2.6.14 vue-templatecompiler@2.6.14

vite

说明:
 Vite 需要 Node.js 版本 18+, 20+。当您的包管理器发出警告时,请注意升级您的 Node 版本,详情请参考 Vite 官 网。

使用 vite 方式创建项目,按照下图选项配置 Vue + TypeScript 。

```
npm create vite@latest
```

Project name: ai-desk-example
Select a framework: Vue
Select a variant:

之后切换到项目目录,安装项目依赖:

cd ai-desk-example npm install

安装插件所需 sass 环境依赖:

npm i -D sass sass-loader

清除项目默认的样式,避免样式问题:

macOS 端



echo -n > src/style.css
Windows 端(PowerShell)
Clear-Content -Path src/style.css
Windows 端(CMD)
echo. > src\style.css

# 步骤2:下载 UI 组件

通过 npm 方式下载 UI 组件,并将 UI 组件复制到自己工程的 src 目录下:

macOS 端
npm i @tencentcloud/ai-desk-customer-vue@latest

mkdir -p ./src/ai-desk-customer-vue && rsync -av --exclude=
 {'node\_modules','excluded-list.txt'} ./node\_modules/@tencentcloud/ai-desk-customer vue/ ./src/ai-desk-customer-vue
Windows 端

npm i @tencentcloud/ai-desk-customer-vue@latest

xcopy .\node\_modules\@tencentcloud\ai-desk-customer-vue .\src\ai-desk-customer-vue
/i /e /exclude:.\node\_modules\@tencentcloud\ai-desk-customer-vue\excluded-list.txt



## 步骤3:引入 UI 组件

#### 在需要展示的页面,引入 UI 的组件即可使用。

#### 例如:在 App.vue 页面中实现以下代码,即可快速搭建客服咨询界面(以下示例代码同时支持 Web 端与 H5 端):

#### () 说明:

以下示例代码使用了 setup 语法,如果您的项目没有使用 setup 语法,请按照 Vue3/Vue2 的标准方式注册组件。

vue3

```
<template>
    </template>
    </template>
    </template=?(width: '600px', height: '80vh', margin: '10px auto', boxShadow: '0
    11px 20px #ccc' }"
    />
    </template>
    </te>
    </template>
    </template>
    </template>
    </
```

#### vue2.7

<pre></pre>
<div id="app"></div>
<customerservicechat< th=""></customerservicechat<>
<pre>:style="{ width: '600px', height: '80vh', margin: '10px auto', boxShadow: '0</pre>
11px 20px #ccc' }"
<script lang="ts" setup=""></script>



```
onMounted(() => {
  const SDKAppID = 0; // Your SDKAppID
  const userID = ''; // Your userID
  const userSig = ''; // Your userSig
  TUICustomerServer.init(SDKAppID, userID, userSig);
});
</script>
</style lang="scss">
</style lang="scss"></style>
```

#### vue2.6及以下

```
<template>

<div id="app">

<CustomerServiceChat

:style="{ width: '600px', height: '80vh', margin: '10px auto', boxShadow: '0

11px 20px #ccc' }"

/>

</div>

</div</div

</div

</di
```

1. 在 main.ts/main.js 中引入 VueCompositionAPI。

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

2. 在 vue.config.js 中增加,若没有该文件请新建。

```
const ScriptSetup = require("unplugin-vue2-script-setup/webpack").default;
module.exports = {
```





# 步骤4:获取 SDKAppID、userID、 userSig

设置 App.vue 中的 SDKAppID 、 userID 、 userSig 。

• SDKAppID 信息,可在即时通信 IM 控制台 单击 应用管理 > 创建新应用,并选择智能客服 Desk,开通智能客服 后获取。

	功能服务							
	⑦ 消息服务 Cha	at 🗸 •						
	☞ 推送服务 Pus	sh ~	智能	能客服 Desk	Hot			
	① 智能客服 Des	sk						
	□ 音视频服务 R	тс	E AI II 度客	《迷学习和楔拟资深各版、 》户关系;	销售刈话,即时督能!	业合,证人工专注复	<del>然</del> 问题处理,打 <b>逗</b> 深	
	通用工具		• 多语	言智能适配,全球多渠道	无缝对接,打破地域刚	<b>艮制,业务拓展无边</b> 级	界。	
	◎ 回调配置		_					
	음 UserSig生成	校验	免	☆ 前 前 前 前 前 前 前 前 前 前 前 前 前 前 前 前 前 前 前				
			<del>开</del> :	通试用				
6。应用管理	应用管理 IM技术展生	<sub>务交流</sub> ッ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・						
<b>己</b> 应用管理 功能服务	应用管理 IM 技术服装 创建航应用	<sub>第交源</sub> 》。 1.复制	I SDKAppID					
<mark>뎒 应用管理</mark> 功能服务 ② <b>消息服务</b> Chat ✓●	<b>应用管理 M 技术服</b> の 現 の 思 の の 日 を 本服 の の た 本服 の の の の た 本服 の の の の の の の の の の の の の	<sup>8交添新</sup> 在 1.复制 SDKAR		应用状态	影響中赴公	又标签①	的總計詞	网络信息
<ul> <li>              G 应用管理             功能服务             G             消息服务 Chat ~・</li></ul>	<b>应用管理 IM 技术服</b> <b>创建新应用</b> 应用名称	<sup>新交通新</sup> 在一部 1.复制 SDKApi	<b>SDKAppID</b>    <b>SD</b> (名称)	应用状态	数据中心①	五 松藻①	¢88#at/m)↓	服务信息
<ul> <li>□ 应用管理</li> <li>□ 消息服务 Chat ···</li> <li>□ 消息服务 Chat ···</li> <li>□ 推送服务 Push ···</li> <li>○ 智能客服 Desk</li> </ul>	<b> 広用管理 M 技术服</b>	<sup>8交演》</sup> 在 1.复制 SDKAP	IJ SDKAppID ₽D⊙ ₩₩	应用状态	数据中心①	五 触靈①	691887140 L	服务信息 消息服务 Chat 体验版 ► 推送服务 Duch ►

• userID 信息,可本地生成一个随机的字符串,例如 test-1234。

• userSig 信息,可单击 即时通信 IM 控制台 > UserSig生成校验,填写创建的 userID,即可生成 userSig。


已 应用管理	UserSig生成&校验 16 现 > 当前数据中心:中国 ① M 技术服务交流群 2	IM 出海交流专区
功能服务		
⑦ 消息服务 Chat ~	签名(UserSig)生成工具	登录鉴权介绍 🖸
☞ 推送服务 Push ~	此工具可以快速生成签名(UserSig),用于本地跑通 Demo 以及功能调试。	
	用户名 (UserID) test-1234 •	
Ⅰ 音视频服务 RTC	密钥	
通用工具	<u>3.填入创建的 userID</u> ************************************	
☆ 回调配置		
名 UserSig生成校验		
☴ 自助排障日志		
	如有需要,请到应用基础信息中复制密钥	
	生成签名(UserSig)	
2.UserSig 生成	校验	
	(UserSig)	
	4.生成签名并复制	
	复制签名 (UserSig)	

# 步骤5:启动项目,发起您的第一条客服咨询

执行以下命令启动项目:

() ·	。" 说明:
:	由于 vue-cli 默认开启 webpack 全局 overlay 报错信息提示,为了您有更好的体验, <b>建议您关闭全局 overlay 报错提</b> <b>示。</b> 在 vue.config.js 中添加以下代码
	webpack4及以上
	<pre>module.exports = defineConfig({</pre>
	devServer: {
	client: {
	overlay: false,
	},
	<pre>}, });</pre>
	webpack3
	•



<pre>module.exports = {    devServer: {       overlay: false,    },   };</pre>	
在tsconfig.json中关闭 ai-desk-customer-vue 的ts检测。 {     "exclude": [     "node_modules",     "src/ai-desk-customer-vue",     ] }	
npm run serve vite	
npm run dev	

# 高级特性

# 国际化界面语言

#### UIKit 支持以下界面语言:

语言代码(userLang)	语言
zh_cn	简体中文
en	英文
zh_tw	繁体中文
ја	日语
id	印尼语
ms	马来语
vi	越南语



th	泰语
fil	菲律宾语
ru	俄语

如果您的业务需要出海,且用户语言以英语为主,可在引入智能客服时设置 userLang="en" 。如果您不指定 userLang,UIKit 会使 用浏览器设置的语言。

<template></template>
<customerservicechat< td=""></customerservicechat<>
<pre>:style="{ width: '600px', height: '80vh', margin: '10px auto', boxShadow: '0 11px</pre>
20px #ccc' }"
userLang="en"

如果您需要支持动态切换用户语言,可使用 TUICustomerServer.changeLanguage 接口,并通过切换 页面/组件 key 的方式,实现 语言动态修改与展示。

<template>
<CustomerServiceChat
:style="{ width: '600px', height: '80vh', margin: '10px auto', boxShadow: '0 11px
20px #ccc' }"
:key="locale"
:userLang="locale"
/>
</template>
<script setup lang="ts">
import TUICustomerServer, { CustomerServiceChat } from './ai-desk-customer-vue';
import { onMounted, ref } from "vue";
const locale = ref('en');
const changeLanguage = (language: string) => {
TUICustomerServer.changeLanguage(language).then(() => {
 locale.value = language;
 });
}
</script>
</style>

#### 用户端带昵称和头像登录

如果人工客服在工作台接待用户咨询时,希望能看到用户的昵称、头像等信息以提升沟通效率,效果如下图所示:



我的	$\oplus$	张三 1852010**** @服务号		& 转接 🕛 结束	用户信息	
<ul> <li>○ 张三 1852010****</li> <li>●服务号</li> <li>▲: 換入成功, 超级管理员为您服务</li> </ul>	网间		REALING HIT RES I HTH RIN	administrator 2025-04-23 15:01:54 会话已结束	用户ID 用户昵称	m9tkv2j6-kutf-2n 张三 1852010**** 昭冬号
		张三 1852010**** 2025-04-23 15:01:58 人工客服				

请使用 initWithProfile 接口初始化,传入昵称和头像即可。

TUICustomerServer.initWithProfile({ SDKAppID.
SDKAppID.
userID,
userSig,
nickName: '张三 1852010****',
<pre>avatar: 'https://im.sdk.qcloud.com/download/tuikit-resource/avatar/avatar_3.png'</pre>

# 工具栏快捷按钮

如果您想实现输入框上方增加快捷按钮,方便用户使用,例如增加"人工客服","发送订单消息"等,可在引入智能客服时设置 toolbarButtonList 。效果如下所示:



toolbarButtonList 是一个包含了一个或多个配置的数组,配置的描述如下:

参数	类型	是否必填	说明
title	String	Yes	button标题。
icon	String	No	button 图标 url。
isPreset	Number	Yes	<ul> <li>● 1:客服组件内置功能。</li> <li>● 0:非内置功能。</li> </ul>



presetId	String	No	当 isPreset 为 1 时,可选值如下: <ul> <li>humanService:人工服务。</li> <li>serviceRating:服务评价。</li> <li>endHumanService:结束对话。</li> </ul>
isEnable d	Number	Yes	● 1: 渲染。 ● 0: 不渲染。
content	String	No	当 isPreset 为 0 时,填入文本内容或者 url。
type	Number	Yes	<ul> <li>1:点击 button 后客服组件发送 content 对应的文本。</li> <li>2:点击 button 后客服组件打开 content 对应的 url(小程序打开 url可能会受限,请提前参考相关文档解决)。</li> </ul>

<template>

```
<CustomerServiceChat
```

```
:style="{ width: '600px', height: '80vh', margin: '10px auto', boxShadow: '0 11px
20px #ccc' }"
```

:toolbarButtonList="toolbarButtonList"

```
/>
```

```
</template>
```

```
<script setup lang="ts">
```

```
import TUICustomerServer, { CustomerServiceChat } from './ai-desk-customer-vue';
import { onMounted } from 'vue';
```

```
const toolbarButtonList = [
```

{"title":"人工服务","icon":"https://tccc-im-agent-avatar-1258344699.cos.apnanjing.myqcloud.com/toolbar\_button\_1.png","type":1,"content":"","isPreset":1,"presetId" :"humanService","isEnabled":1},

```
{"title":"服务评价","icon":"https://tccc-im-agent-avatar-sg-1258344699.cos.ap-
singapore.myqcloud.com/toolbar_button_2.png","type":1,"content":"","isPreset":1,"presetI
d":"serviceRating","isEnabled":1},
```

{"title":"结束对话","icon":"https://tccc-im-agent-avatar-sg-1258344699.cos.apsingapore.myqcloud.com/toolbar\_button\_3.png","type":1,"content":"","isPreset":1,"presetI d":"endHumanService","isEnabled":1},

```
{"type":1,"title":"智能客服","content":"智能客服","isPreset":0,"isEnabled":1}
];
```

```
onMounted(() => {
   const SDKAppID = 0; // Your SDKAppID, 即开通了智能客服 Desk 的应用 ID
   const userID = ''; // Your userID, 可复用您 app 的账号体系,或随机生成
   const userSig = ''; // Your userSig, 接入阶段可控制台生成, 生产阶段请务必由服务端生成
   TUICustomerServer.init(SDKAppID, userID, userSig);
});
</script>
</style scoped>
</style>
```

#### 底部快捷订单

# () 说明:

腾讯云

请升级 ai-desk-customer-vue 到 v1.5.2 或更高版本。

如果您想实现打开客服会话时在聊天区域底部展示快捷订单,可以配置 bottomQuickOrder ,效果如下所示:



bottomQuickOrder 的参数描述如下:

参数	类型	是否必填	说明
header	String	Yes	订单标题,对应上图的"智能客服高级版"。
desc	String	Yes	订单描述,对应上图的"3000/月"。
pic	String	No	订单图片 url。
url	String	Yes	当客服坐席收到订单消息时,点击可打开此 url 对应的页面。
customF ield	Array	No	自定义配置,例如"订单号","订单时间"等信息。

template> <CustomerServiceChat :style="{ width: '600px', height: '80vh', margin: '10px auto', boxShadow: 20px #ccc' }"

:bottomQuickOrder="bottomQuickOrder"



```
header: "高级版智能客服(包含3个客服许可)",
desc: "¥3000/月",
   name: '订单号',
   name: '下单时间',
const SDKAppID = 0; // Your SDKAppID, 即开通了智能客服 Desk 的应用 ID
const userID = ''; // Your userID, 可复用您 app 的账号体系, 或随机生成
const userSig = ''; // Your userSig, 接入阶段可控制台生成, 生产阶段请务必由服务端生成
```

# 多客服号

 说明: 请升级 ai-desk-customer-vue 到 v1.5.3 或更高版本。

如果您的应用内需要多个客服号,为用户提供专属客服功能,以提高服务质量和响应速度,可用 UIKit 提供的多客服号功能实现。效果如 下所示:







```
// 1. 初始化前将业务侧自定义的客服号 ID 传递给 UIKit
// 自定义客服号请使用 restapi = https://cloud.tencent.com/document/product/269/111798
TUICustomerServer.setCustomerServiceIDList(['customerServiceID_001',
'customerServiceID_002', 'customerServiceID_003']);
// 2. 初始化时,带上要发起会话的客服号 ID,比如 'customerServiceID_002'
// 如果不带上要发起会话的客服号 ID,则 UIKit 默认使用多客服号列表的第 1 个元素
TUICustomerServer.initWithProfile({
SDKAppID,
userID,
userSig,
nickName: '张三 1562010****',
avatar: 'your avatar url',
customerServiceID: 'customerServiceID_002',
});
```

# 用户端主动结束人工会话

# 🕛 说明:

用户端可以通过发送自定义消息的方式实现主动结束会话,适用于以下 3 种情况:

- 1. 用户转人工触发排队,发送此消息可以结束排队。
- 2. 客服接待方式为手动接待,用户转人工分配客服成功后等待客服确认接待,发送此消息可以结束等待。
- 3. 用户转人工且成功接入人工客服,发送此消息可以结束本次会话。

```
IUICustomerServer.sendCustomMessage(-
   to: '@customer_service_account',
   conversationType: 'C2C',
   payload: {
```





#### 常见问题

#### 什么是 UserSig? 如何生成 UserSig?

UserSig 是用户登录即时通信 IM 的密码,其本质是对 UserID 等信息加密后得到的密文。 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端,并提供面向项目的接口,在需要 UserSig 时由您的项目向业务服务器 发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig 。

#### JS 工程如何接入 TUIKit 组件?

UIKit 仅支持 ts 环境运行,您可以通过渐进式配置 typescript 来使您项目中已有的 js 代码 与 UIKit 中 ts 代码共存。

vue-cli

请在您 vue-cli 脚手架创建的工程根目录执行:

vue add typescript

之后按照如下进行配置项进行选择(为了保证能同时支持原有 js 代码 与 UIKit 中 ts 代码,请您务必严格按照以下五个选项进行配



#### 完成以上步骤后,请重新运行项目!

Vite

请在您 vite 创建的工程根目录执行:

pm install -D typescript

运行报错: Uncaught TypeError: marked\_\_\_WEBPACK\_IMPORTED\_MODULE\_0\_ Marked is not a constructor

如果您运行过程中出现如下错误,说明您当前 Vue CLI 构建的项目环境版本较低,需要降低 uikit 中使用的 marked 版本至 5.1.2。



Our caught TypeError: marked_WEBPACK_IMPORTED_MODULE_0Marked is not a constructor at eval (marked_ts:22:1)	marked.ts:22
at /src/TIIKit/components/TIIIChat/massage_list/massage_elements/massage_stream_markdown/marked ts (app	ic.8381.1)
at , sic/ totki/ components/ totki/ message=tis/ message=tis/ message=tis/ message=tis/ message=tis/	<u>)3.0301.1</u> )
atwebpack_reduite ( <u>app.js:o49:30</u> )	
at th ( <u>app.)s:151:20</u> )	
at ./node_modules/cache-loader/dist/cjs.js?!./node_modules/babel-loader/lib/index.js!./node_modules/ts-	
loader/index.js?!./node_modules/cache-loader/dist/cjs.js?!./node_modules/vue-	
<pre>loader/lib/index.js?!./node_modules/unplugin/dist/webpack/loaders/transform.js?unpluginName=unplugin-vue2-sc</pre>	ript–
setup!./src/TUIKit/components/TUIChat/message-list/message-elements/message-stream-markdown/index.vue?	
vue&type=script⟨=ts ( <u>app.js:1310:1</u> )	
atwebpack_require ( <u>app.js:849:30</u> )	
at fn (app.js:151:20)	
at eval ( <u>index.vue:1:1</u> )	
at ./src/TUIKit/components/TUIChat/message-list/message-elements/message-stream-markdown/index.vue?	
vue&type=script⟨=ts ( <u>app.js:8357:1</u> )	
<u>显示已列入忽略列表的帧</u>	

请在您项目的 根目录 使用以下脚本降低 marked 版本:

#### npm i marked@5.1.2 --legacy-peer-deps

编译报错 node\_modules/marked/lib/marked.esm.js: Class private methods are not enabled.

如果您运行过程中出现如下错误,说明您当前使用的 marked 版本过低,请升级 marked 版本至 6.0.0。



请在您项目的根目录使用以下脚本升级 marked 版本:

npm i marked@6.0.0 --legacy-peer-deps



# uni-app

最近更新时间: 2025-07-11 15:02:32

# 介绍

智能客服用户端 uni–app UlKit。使用此 UlKit,您可以在一天内将智能客服的能力集成到您的 H5、小程序、App 项目。极简接入, 一套代码多端运行且体验一致,用智能客服为您的产品增收提效,提升用户满意度和付费转化。

# 效果展示



# 开发环境要求

- HBuilderX 升级到最新版本
- TypeScript / JavaScript (UIKit 使用 ts 语言开发,支持在 js 或者 ts 项目中集成)
- Vue2 / Vue3
- sass (sass-loader 版本 ≤ 10.1.1)
- node(12.13.0 ≤ node 版本 ≤ 17.0.0, 推荐使用 Node.js 官方 LTS 版本 16.17.0)
- npm (版本请与 node 版本匹配 )

# UIKit 源码集成



## 步骤1: 创建项目(已有项目可忽略)

打开 HbuilderX,在菜单栏中选择"文件-新建-项目",创建一个名为 ai-desk-example 的 uni-app 项目。Vue 版本选择推荐 3。

	ai-desk-example				
/ap2App	D:/dev/desk/development				1
5+App	选择模板			Q 查找模板	去搜索
DE插件		Image: Second	Hello uvue	Hellouni-app           XirryiniaapyEr/2005/04/14	
			注道 	<ul> <li>□. 模板, 逆信</li> <li>□. 個ut-d</li> <li>型場整合</li> <li>型機能の構築</li> </ul>	Var, Var           Vier problem           Vier problem
	<b>Hello uts</b> uts语法开发原生示例, <u>详</u> 情	uni-starter 云端一体应用快速开发基本 项目横版,注情	<b>uni-admin</b> admin管理系统基础框架, 详情	<b>uni-ui项目</b> 基于uni-app的 UI 组件库, 详情	<b>Hello i18n</b> uni-app 国际化演示,详情

# 步骤2: 下载 UIKit

• HBuilderX 创建项目时默认不会创建 package.json 文件,请在项目根目录下执行以下命令创建 package.json 文件:





🔗 腾讯云

sync -av --exclude={'node\_modules','excluded-list.txt'}

./node\_modules/@tencentcloud/ai-desk-customer-uniapp/ ./pages-ai-desk/ai-deskcustomer-uniapp/

#### Windows 端

xcopy .\node\_modules\@tencentcloud\ai-desk-customer-uniapp .\pages-ai-desk\ai-deskcustomer-uniapp /i /e /exclude:.\node\_modules\@tencentcloud\ai-desk-customeruniapp\excluded-list.txt

# 步骤3:引入 UIKit

#### 1. 工程配置

```
manifest.json 文件
```

在 manifest.json 文件的源码视图中开启小程序分包 subPackages 和关闭 H5 treeShaking 选项。

```
// weixin miniProgram
"mp-weixin" : {
    "appid" : "",
    "optimization" : {
        "subPackages" : true
    }
},
// H5: close treeshaking to solve the problem of uni[methond]() is not a function
"h5" : {
        "optimization" : {
            "treeShaking" : {
               "enable" : false
            }
        },
        // M2序默认使用分包集成, 打包小程序时 manifest.json 不要配置 lazyCodeLoading 选项。
```

vue.config.js(Vue2项目请修改;Vue3项目无需修改)

Vue2项目必须在根目录下创建或修改 vue.config.js 。



<pre>const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;</pre>
<pre>module.exports = {</pre>
parallel: false,
configureWebpack: {
plugins: [
ScriptSetup({
/* options */
<pre>}),</pre>
],
},
chainWebpack(config) {
// disable type check and let `vue-tsc` handles it
<pre>config.plugins.delete('fork-ts-checker');</pre>
},
};

#### 2. 分包集成 UIKit

为了规避小程序主包体积超限问题,我们建议您将客服 UIKit 放入分包。假设您的主包页面是 pages/index/index.vue ,分包页 面是 pages-ai-desk/index/index.vue ,主包页面有一个咨询客服的入口,用户点击后跳转至分包的智能客服页面。项目的目 录结构如下所示:

```
----YOUR-UNI-APP-PROJECT
----pages
-----index
----pages-ai-desk
-----pages-ai-desk
-----ai-desk-customer-uniapp
-----index
-----index.vue
----App.vue
-----manifest.json
-----pages.json
```

#### 请将以下内容复制到项目对应的文件中。

#### App.vue 文件

```
<script>
export default {
    onLaunch: function() {
        console.log('App Launch')
    },
    onShow: function() {
        console.log('App Show')
    },
```



<pre>onHide: function() {</pre>
<pre>console.log('App Hide')</pre>
<style></td></tr><tr><td>uni-page-body,</td></tr><tr><td>html,</td></tr><tr><td>body,</td></tr><tr><td>page {</td></tr><tr><td>width: 100% !important;</td></tr><tr><td>height: 100% !important;</td></tr><tr><td>overflow: hidden;</td></tr><tr><td></td></tr><tr><td>#app {</td></tr><tr><td>height: 100% !important;</td></tr><tr><td></td></tr><tr><td></style>

#### pages.json 文件



```
"current": 0, //当前激活的模式(list 的索引项)
"list": [{
    "name": "", //模式名称
    "path": "", //启动页面,必选
    "query": "" //启动参数,在页面的onLoad函数里面得到
    }]
}
```

main.js(Vue2项目请修改;Vue3项目无需修改)

如果您是 Vue2 项目,请在 main.js 中引入组合式API,防止环境变量 isPC 等无法使用。



#### 3. 在项目中配置智能客服聊天的入口

#### 主包 pages/index/index.vue

```
<template>
<template>
<div class="chat">
<button @click="getSupport">咨询客服</button>
</div>
</template>
<script>
export default {
    methods: {
        getSupport() {
            uni.navigateTo({
               url; "/pages-ai-desk/index/index"
            })
        }
    }
</script>
<style lang="scss" scoped>
.chat {
    height: 100%;
    overflow: hidden;
    display: flex;
    align-items: center;
    justify-content: center;
}
```



#### /style>

#### 分包 pages-ai-desk/index/index.vue

```
// SDKAppID/userID/userSig 的获取请参考步骤4
const SDKAppID = 0; // Your SDKAppID,即开通了智能客服 Desk 的应用 ID
const userID = ''; // Your userID, 可复用您 app 的账号体系, 或随机生成
const userSig = ''; // Your userSig, 接入阶段可控制台生成, 生产阶段请务必由服务端生成
// 带昵称和头像登录
```

#### 步骤4:获取 SDKAppID 、userID 、 userSig

设置 pages-ai-desk/index/index.vue 中的 SDKAppID 、 userID 、 userSig 。

• SDKAppID 信息:在 即时通信 IM 控制台 单击 应用管理 > 创建新应用,并选择智能客服 Desk, 开通智能客服 后获取。



功能服务	
⑦ 消息服务 Chat ✓●	
☞ 推送服务 Push   ~	智能客服 Desk 🔤
♀ 智能客服 Desk	• AI 快速学习和模拟资深客服、销售对话,即时智能应答。让人工专注复杂问题处理,打造深
□ 音视频服务 RTC	度客户关系;
通用工具	•多语言智能适配,全球多渠道无缝对接,打破地域限制,业务拓展无边界。
☆ 回调配置	
名 UserSig生成校验	免疫运用 工即购失
	开通试用

<b>己</b> 应用管理	应用管理 Ⅲ 技术服务							
功能服务	创建新应用	I.复利 JUK	нррго					
⑦ 消息服务 Chat ~•	应用名称		家钼	应用状态	数据中心①	☑ 标签①	創建时间 」	服条信息
☞ 推送服务 Push ~	122.713 124.101	obiotppin ()	11 IN	2227131070104	xxxx:1.10-()	i matt	63AE+31-7 ¥	AN EL CC 201
智能客服 Desk								消息服务 Chat 体验版 ▶
国 音视频服务 RTC	qiaoqiao1111	e	© Þ	正常	中国	-	2025-06-27	推送服务 Push ▶
								智能客服 Desk 高级版试用 I

- userID 信息,可本地生成一个随机的字符串,例如 test-1234。
- userSig 信息,可单击 即时通信 IM 控制台 > UserSig生成校验,填写创建的 userID,即可生成 userSig。

🔁 应用管理	UserSig生成&校验 16 現 > 当前数据中心:中国① IM 技术服务交流群 IC	IM 出海交流专区
功能服务		
⑦ 消息服务 Chat ~	签名(UserSig)生成工具	登录鉴权介绍 已
☞ 推送服务 Push ~	此工具可以快速生成签名(UserSig),用于本地跑通 Demo 以及功能调试。	
	用户名(UseriD) test-1234 •	
■ 音视频服务 RTC	密钥	
通用工具	3.填入创建的 userID	
◎ 回调配置		
名 UserSig生成校验		
⇒ 自助排障日志		
	如有需要,请到应用基础信息中复制密钥	
	牛成茶名 (UserSia)	
2.UserSig 生成	<i>就校验</i>	
	4.生成签名并复制	
	复制签名(UserSig)	

# 步骤5:启动项目,并发起您的第一条客服咨询

- 1. 使用 HBuilderX 启动该项目,单击运行,可选择运行到浏览器,或者手机,或者小程序模拟器。
- 2. 如果您选择了运行到微信开发者工具,但 HBuilderX 没有自动拉起微信开发者工具,请使用微信开发者工具手动打开编译后的项目, 目录地址: unpackage/dist/dev/mp-weixin 。
- 小程序开发环境,请选择 详情 > 本地设置中勾选 不校验合法域名、web−view(业务域名)、TLS版本以及 HTTPS 证书。上线前请在微信公众平台 > 开发 > 开发管理 > 开发设置 > 服务器域名中进行域名配置,域名配置详见:小程序合法域名。



# 高级特性

#### 国际化界面语言

UIKit 支持以下界面语言:

语言代码(userLang)	语言
zh_cn	简体中文
en	英文
zh_tw	繁体中文
ја	日语
id	印尼语
ms	马来语
vi	越南语
th	泰语
fil	菲律宾语
ru	俄语

如果您的业务需要出海,且用户语言以英语为主,可在引入智能客服时设置 userLang="en" 。如果您不指定 userLang,UIKit 会使 用浏览器或 App 设置的语言。

```
<template>

<CustomerServiceChat style="height: 100%;"

userLang="en"

/>

</template>
```

如果您需要支持动态切换用户语言,可使用 TUICustomerServer.changeLanguage 接口,并通过切换 页面/组件 key 的方式,实现 语言动态修改与展示。

```
<template>

<CustomerServiceChat style="height: 100%;"

:key="locale"

:userLang="locale"

/>

</template>

<script>

import TUICustomerServer from '../ai-desk-customer-uniapp';

import CustomerServiceChat from '../ai-desk-customer-

uniapp/components/CustomerServiceChat/index-uniapp.vue';

export default {

components: {
```



CustomerServiceChat,
<pre>let locale = 'en';</pre>
locale
methods: {
changeLanguage (language) {
TUICustomerServer.changeLanguage(language).then(() => {
this.locale = language;
<style lang="scss" scoped=""></td></tr><tr><td></style>

#### 用户端带昵称和头像登录

如果您希望人工客服在工作台接待用户咨询时,能看到用户的昵称、头像等信息以提升沟通效率,效果如下所示:

我的	$\oplus$	张三 1852010**** @服务号		& 转接 ● 结束	用户信息	
<ul> <li>○ 张三 1852010****</li> <li>●服务号</li> <li>▲:接入成功,超级管理员为您服务</li> </ul>	RURU		派CRANINH/- 农区 1 计加速用	administrator 2025-04-23 15:01:54 会话已结束	用户ID 用户昵称	m9tkv2j6-kutf-2n 张三 1852010****
		強三 1852010**** 2025-04-23 15:01:58 人工客服			216.215	ר כלאמ

请使用 initWithProfile 接口初始化,传入昵称和头像即可。

TUICustomerServer.initWithProfile({	
SDKAppID,	
userID,	
userSig,	
nickName: '张三 1852010****',	
<pre>avatar: 'https://im.sdk.qcloud.com/download/tuikit-resource/avatar/avatar_3.png'</pre>	

# 工具栏快捷按钮

如果您想实现输入框上方增加快捷按钮,方便用户使用,例如增加"人工客服","发送订单消息"等,可在引入智能客服时设置 toolbarButtonList 。效果如下所示:





toolbarButtonList 是一个包含了一个或多个配置的数组,配置的描述如下:

参数	类型	是否必填	说明
title	String	Yes	button标题。
icon	String	No	button 图标 url。
isPreset	Number	Yes	<ul> <li>1:客服组件内置功能。</li> <li>0:非内置功能。</li> </ul>
presetId	String	No	当 isPreset 为 1 时,可选值如下: <ul> <li>humanService:人工服务。</li> <li>serviceRating:服务评价。</li> <li>endHumanService:结束对话。</li> </ul>
isEnable d	Number	Yes	● 1: 渲染。 ● 0: 不渲染。
content	String	No	当 isPreset 为 0 时,填入文本内容或者 url。
type	Number	Yes	<ul> <li>1:点击 button 后客服组件发送 content 对应的文本。</li> <li>2:点击 button 后客服组件打开 content 对应的 url(小程序打开 url 可能会受限,请提前参考相关文档解决)。</li> </ul>

<template>

```
<CustomerServiceChat style="height: 100%;"
:toolbarButtonList="toolbarButtonList"
/>
/template>
script>
```





#### 底部快捷订单

```
    说明:
请升级 ai-desk-customer-uniapp 到 v1.5.2 或更高版本。
```

如果您想实现打开客服会话时在聊天区域底部展示快捷订单,可以配置 bottomQuickOrder ,效果如下所示:





bottomQuickOrder 的参数描述如下:

参数	类型	是否必填	说明
header	String	Yes	订单标题,对应上图的"智能客服高级版"。
desc	String	Yes	订单描述,对应上图的"3000/月"。
pic	String	No	订单图片 url。
url	String	Yes	当客服坐席收到订单消息时,点击可打开此 url 对应的页面。
customF ield	Array	No	自定义配置,例如"订单号","订单时间"等信息。

```
<template>
    </template>
    </template>
    </template>
    </template>
    </template>
    </template>
    import TUICustomerServer from '../ai-desk-customer-uniapp';
    import CustomerServiceChat from '../ai-desk-customer-
uniapp/components/CustomerServiceChat/index-uniapp.vue';

export default {
</template>
```



```
header: "高级版智能客服(包含3个客服许可)",
   name: '订单号',
  name: '下单时间',
```

# 多客服号

```
    说明:
请升级 ai-desk-customer-uniapp 到 v1.5.3 或更高版本。
```

如果您的应用内需要多个客服号,为用户提供专属客服功能,以提高服务质量和响应速度,可用 UIKit 提供的多客服号功能实现。效果如 下所示:





```
// 1. 初始化前将业务侧自定义的客服号 ID 传递给 UIKit
// 自定义客服号请使用 restapi - https://cloud.tencent.com/document/product/269/111798
TUICustomerServer.setCustomerServiceIDList(['customerServiceID_001',
'customerServiceID_002', 'customerServiceID_003']);
// 2. 初始化时,带上要发起会话的客服号 ID,比如 'customerServiceID_002'
// 如果不带上要发起会话的客服号 ID,则 UIKit 默认使用多客服号列表的第 1 个元素
TUICustomerServer.initWithProfile({
SDKAppID,
userID,
userSig,
nickName: '张三 1562010****',
avatar: 'your avatar url',
customerServiceID: 'customerServiceID_002',
});
```

# 用户端主动结束人工会话

# 🕛 说明:

用户端可以通过发送自定义消息的方式实现主动结束会话,适用于以下 3 种情况:

- 1. 用户转人工触发排队,发送此消息可以结束排队。
- 2. 客服接待方式为手动接待,用户转人工分配客服成功后等待客服确认接待,发送此消息可以结束等待。
- 3. 用户转人工且成功接入人工客服,发送此消息可以结束本次会话。

```
IUICustomerServer.sendCustomMessage(-
   to: '@customer_service_account',
   conversationType: 'C2C',
   payload: {
```





# 通过 vue-cli 命令行集成

除了HBuilderX可视化界面,您也可以使用 cli 脚手架,通过 vue−cli 创建 uni−app 项目。这两种方式在集成过程上有所区别,请您参 考下面的步骤进行。

uni-app vue2 默认模板

1. 创建 vue2 项目,使用默认模板。

vue create -p dcloudio/uni-preset-vue my-project

2. 在项目目录 my-project 内下载 UIKit。

npm i @tencentcloud/ai-desk-customer-uniapp@latest unplugin-vue2-script-setup

3. 下载 sass/sass-loader/ts-loader 等。

```
npm i sass sass-loader@10.1.1 ts-loader typescript @vue/cli-plugin-typescript --
legacy-peer-deps
```

4. 将 UIKit 源码复制到项目中。请注意,复制到工程的 src 目录下。

macOS 端

mkdir -p ./src/pages-ai-desk/ai-desk-customer-uniapp &&
rsync -av --exclude={'node\_modules','excluded-list.txt'}
./node\_modules/@tencentcloud/ai-desk-customer-uniapp/ ./src/pages-ai-desk/aidesk-customer-uniapp/

#### Windows 端

xcopy .\node\_modules\@tencentcloud\ai-desk-customer-uniapp .\src\pages-aidesk\ai-desk-customer-uniapp /i /e /exclude:.\node\_modules\@tencentcloud\aidesk-customer-uniapp\excluded-list.txt

#### 5. 在工程目录下新建 vue.config.js 文件。

腾讯云

#### 6. 在工程目录下新建 tsconfig.json 文件。

```
{
  "compilerOptions": {
    "target": "esnext",
    "module": "esnext",
    "strict": true,
    "jsx": "preserve",
    "importHelpers": true,
    "moduleResolution": "node",
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "sourceMap": true,
    "skipLibCheck": true,
    "baseUrl": ".",
    "types": [
        "webpack-env",
        "@dcloudio/types"
    ],
    "paths": {
        "@/*": [
        "./src/*"
      ]
    },
    "lib": [
      "esnext",
      "dom",
      "dom.iterable",
      "scripthost"
```





7. 其它步骤同 HBuilderX 可视化界面创建的项目一致,不再赘述。

uni-app vue2 TypeScript 模板

1. 创建 vue2 项目,使用默认模板(TypeScript)。

vue create -p dcloudio/uni-preset-vue my-project

2. 在项目目录 my-project 内下载 UIKit。

npm i @tencentcloud/ai-desk-customer-uniapp@latest unplugin-vue2-script-setup

3. 下载 sass/sass-loader 等。

npm i sass sass-loader@10.1.1 --legacy-peer-deps

4. 将 UIKit 源码复制到项目中。请注意,复制到工程的 src 目录下。

macOS 端

mkdir -p ./src/pages-ai-desk/ai-desk-customer-uniapp &&
rsync -av --exclude={'node\_modules','excluded-list.txt'}
./node\_modules/@tencentcloud/ai-desk-customer-uniapp/ ./src/pages-ai-desk/
desk-customer-uniapp/

Windows 端

xcopy .\node\_modules\@tencentcloud\ai-desk-customer-uniapp .\src\pages-aidesk\ai-desk-customer-uniapp /i /e /exclude:.\node\_modules\@tencentcloud\ai-



#### desk-customer-uniapp\excluded-list.tx

#### 5. 在工程目录下新建 vue.config.js 文件。

6. 编辑 tsconfig.json 文件, exclude UIKit 代码。

```
"exclude": [
    ...
    "src/pages-ai-desk/ai-desk-customer-uniapp"
]
```

7. 其它步骤同 HBuilderX 可视化界面创建的项目一致,不再赘述。

#### uni-app vue3

1. 创建 vue3 项目,用 javascript 开发。

npx degit dcloudio/uni-preset-vue#vite my-project

或创建 vue3 项目,用 typescript 开发。

npx degit dcloudio/uni-preset-vue#vite-ts my-project

2. 在项目目录 my-project 内下载 UIKit。

npm i @tencentcloud/ai-desk-customer-uniapp@latest

3. 下载 sass/sass-loader 等。



5. 其它步骤同 HBuilderX 可视化界面创建的项目一致,不再赘述。

# 消息推送

#### () 说明:

UIKit 中默认没有集成 TencentCloud-TIMPush 推送插件。TencentCloud-TIMPush 是腾讯云即时通信 IM Push 插件。目前推送支持小米、华为、荣耀、OPPO、vivo、魅族、APNs、一加、realme、iQOO 和 苹果等厂商通道。 如果您需要在 App 中集成离线推送能力,请参见 uni-app 推送 实现。





# 常见问题

# 什么是 UserSig? 如何生成 UserSig?

UserSig 是用户登录即时通信 IM 的密码,其本质是对 UserID 等信息加密后得到的密文。

UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端,并提供面向项目的接口,在需要 UserSig 时由您的项目向业务服务器 发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig 。

# 集成 UIKit 在 page.json 中默认没有配置 tabBar,项目中如何实现 tabBar 功能?

如果您打包 App / H5,需要在 pages.json 中配置 tabBar。请参考 uni−app 官网 tabBar 配置 自实现。 如果您打包小程序,因为主包体积限制,小程序默认是分包集成,如果您的 tabBar 需要自定义实现。请参考 uni−app 官网 自定义 tabBar 。

# 小程序如果需要上线或者部署正式环境怎么办?

请在 微信公众平台 > 开发 > 开发管理 > 开发设置 > 服务器域名 中进行域名配置。域名配置详见: 小程序合法域名。



# 微信小程序

最近更新时间: 2025-07-11 15:02:32

# 介绍

智能客服用户端的微信小程序 UIKit。使用此 UIKit,您可以在一天内将智能客服的能力集成到您的小程序项目。极简接入,用智能客服 为您的产品增收提效,提升用户满意度和付费转化。

# 效果展示



# 开发环境要求

- 微信开发者工具
- JavaScript 或 TypeScript
- node (node.js  $\ge$  16.0.0)
- npm(版本请与 node 版本匹配)

# UIKit 源码集成

# 步骤1: 创建项目

在微信开发者工具上创建一个使用 JS-基础模板的小程序项目(是否使用模板,是否使用 TS 均可),如图所示:





# 步骤2: 下载 UIKit

微信开发者工具创建的小程序项目默认没有 package.json,因此您需要先创建 package.json 文件。



通过 NPM 方式下载 UIKit。

npm i @tencentcloud/ai-desk-customer-wechat@latest

为了方便您对 UI 进行扩展,请在项目的根目录下执行以下命令,将 UIKit 源码复制到项目中。





🔗 腾讯云

robocopy .\node\_modules\@tencentcloud\ai-desk-customer-wechat .\ai-desk-customerwechat /E

# 步骤3:分包引入 UIKit

为了规避小程序主包体积超限问题,我们建议您将客服 UIKit 放入分包。

#### 1. 工程配置

app.js

说明:
 以下代码使用了分包异步化的技术方案。

"use strict";
<pre>Object.defineProperty(exports, Symbol.toStringTag, { value: "Module" });</pre>
let common_vendor;
<pre>require.async('./ai-desk-customer-wechat/common/vendor').then((res) =&gt; {</pre>
<pre>common_vendor = res;</pre>
const <b>_sfc_main</b> = {
<pre>onLaunch: function() {</pre>
<pre>console.log("App Launch");</pre>
},
<pre>onShow: function() {</pre>
<pre>console.log("App Show");</pre>
},
<pre>onHide: function() {</pre>
<pre>console.log("App Hide");</pre>
}
};
<pre>function createApp() {</pre>
<pre>const app = common_vendor.createSSRApp(_sfc_main);</pre>
return {
app
};
}
<pre>createApp().app.mount("#app");</pre>
<pre>exports.createApp = createApp;</pre>
<pre>}).catch(({ errMsg, mod }) =&gt; {</pre>
<pre>console.error(`path: \${mod}, \${errMsg}`)</pre>
<pre>});</pre>

app.json



() 说明:

# 请务必配置 subPackages和 preloadRule。

# 2. 在项目主包页面中配置智能客服的入口





const SDKAppID = 0;
const userID = '';
const userSig = '';
<pre>wx.navigateTo({</pre>
url: `/ai-desk-customer-wechat/pages/index/index?
SDKAppID=\${SDKAppID}&userID=\${userID}&userSig=\${userSig}`
})
},
})

主包 pages/index/index.wxml

```
<button class="get-support" bind:tap="getSupport">咨询客服</button>
```

```
主包 pages/index/index.wxss
```

```
page {
   height: 100vh;
   display: flex;
   flex-direction: column;
}
.get-support {
   background-color: #0052d9;
   color: #ffffff;
   margin: auto;
}
```

# 步骤4: 获取 SDKAppID 、userID 、 userSig

设置 pages/index/index.js 中的 SDKAppID 、 userID 、 userSig 。
 SDKAppID 信息,可在 即时通信 IM 控制台 单击应用管理 > 创建新应用,并选择智能客服 Desk, 开通智能客服 后获取。


<ul> <li>○ 消息服务 Chat →</li> <li>○ 指総客服 Desk</li> <li>● 音视频服务 RTC</li> <li>□ 通用工具</li> <li>③ 回调配置</li> <li>④ UserSig生成校验</li> </ul>	功能服务	
<ul> <li>▶ 推送服务 Push →</li> <li>● 智能客服 Desk</li> <li>▲ 留能客服 Desk Not</li> <li>▲ 自税頻服务 RTC</li> <li>通用工具</li> <li>④ 回调配置</li> <li>④ UserSig生成校验</li> </ul> <i>Q</i> 数 <i>Q</i> 3 <i>Q</i> 3 <i>Q</i> 3 <i>Q</i> 3 <i>Q</i> 3 <i>Q</i> 3 <i>Q</i> 4 <i>Q</i> 4 <i>Q</i> 4 <i>Q</i> 4 <i>Q</i> 4 <i>Q</i> 4 <i>D</i> 4 <i>Q</i> 5 <i>Q</i> 4 <i>Q</i> 5 <i>Q</i> 4	⑦ 消息服务 Chat ✓●	
<ul> <li>              ● 智能客服 Desk          </li> <li>              音视频服务 RTC              </li> <li>             適用工具             </li> <li>             回调配置              </li> <li>             UserSig生成校验         </li> </ul> <li> <ul> <li>                  く良気は用                  立即购実</li></ul></li>	☞ 推送服务 Push 🗸	智能客服 Desk 🚥
回 音视频服务 RTC     度客户关系;       通用工具     ・多语言智能适配,全球多渠道无缝对接,打破地域限制,业务拓展无边界。       ② 回调配置        品 UserSig生成校验 <u>文即购买</u> <u>开通试用</u>	♀ 智能客服 Desk	• AI 快速学习和模拟资深客服、销售对话,即时智能应答,让人工专注复杂问题处理,打造深
通用工具     • 多语言智能适配,全球多渠道无缝对接,打破地域限制,业务拓展无边界。       ③ 回调配置	□ 音视频服务 RTC	度客户关系;
<ul> <li>◇ 回调配置</li> <li>2.8 UserSig生成校验</li> <li> <i>免</i>度试用         立即购买 <i>工</i>即购买 <i>开通</i>试用         </li> </ul>	通用工具	•多语言智能适配,全球多渠道无缝对接,打破地域限制,业务拓展无边界。
22 UserSig生成校验	◎ 回调配置	
开通试用	名 UserSig生成校验	免疫证用 立即购买
开通试用		
		开通试用

<b>己</b> 应用管理	应用管理 IM 技术服务								
功能服务	创建新应用	I.复利 SDK	нррій						
⑦ 消息服务 Chat ✓● ⑦ 推送服务 Push ✓	应用名称	SDKAppID	密钥	应用状态	数据中心(i)	7 1	标签门	创建时间 ↓	服务信息
○ 智能客服 Desk					+ <b>P</b>				消息服务 Chat 体验版 ►
□ 音视频服务 RTC	qiaoqiao1111	Ę.	© P	止芾	甲国	-	-	2025-06-27	/推达服务 Push ▶ 智能客服 Desk 高级版试用 ▶

- userID 信息,可本地生成一个随机的字符串,例如 test-1234。
- userSig 信息,可单击 即时通信 IM 控制台 > UserSig生成校验,填写创建的 userID,即可生成 userSig。

🔁 应用管理	UserSig生成&校验 16 现 > 当前数据中心:中国 ① IM 技术服务交流群 I2 IM 出海交流专区
功能服务	
⑦ 消息服务 Chat ~	签名(UserSig)生成工具 登录鉴权介绍 IC
☞ 推送服务 Push ~	此工具可以快速生成签名(UserSig),用于本地跑通 Demo 以及功能调试。
◎ 客服服务 Desk ∨	用户名 (UserID) test-1234 •
国 音视频服务 RTC	密钥
通用工具	3.填入创建的 userID
☆ 回调配置	
名 UserSig生成校验	
☴ 自助排障日志	
	如有需要,请到应用基础信息中复制密钥
	生成签名 (UserSig)
2.User5ig 生成	校验 Alteration
	(UserSig)
	4.生成签名并复制
	新知知名 (Itenetia)
	复初选者(Usersig)

# 步骤5:启动项目,并发起您的第一条客服咨询

测试前,请先清理微信开发者工具的缓存,并勾选**本地设置 > 不校验合法域名**。



基本信	息 性能质量	量 本地设置	项目配置
调试	基础库⑦	3.7.10 45.85%	• 推送
> <del>&gt;</del>			
iOS	山牛又行(2016)合广)响		8.0.55 及以上版本
Andro	oid		8.0.55 及以上版本
MacC	C		暂不支持
Wind	OWS		暂不支持
_	白田冬供信汉		
	何乃漏⊫成[3] 庙田 SMC 编译脚本	文件	
	に用 SVVC 編 単脚本。		
	#1# WOIKIEL   대리	trà là	
· · · ·	L151\19911두I\日4		
	上传(1571)1111111111111111111111111111111111	御中,又作	
		6WXMI,又1+	
-	上传时过滤无依赖又	L14 ۲	
	上传时进行代码保护	-	
	目动运行体验评分		
<b>V</b>	不伦验台法喊名、w 本以及 HTTPS 证书	/eb-view (业务域省	i)、ILS 和文
	预览及真机调试时主	E包、分包体积上限	调整为4M
	启用数据预拉取		
<b>_</b> )	启用代码自动热重载	Ż	
	开启 Skyline 渲染调	试	
' <b>1±</b>	启用多核心编译		
出呢我		<b>.</b>	
IP WERD	州大隊豆	*	

如果您希望人工客服在工作台接待用户咨询时,能看到用户的昵称、头像等信息以提升沟通效率,效果如下所示:

我的	$\oplus$	张三 1852010**** @服务号		𝔅 转接 🕛 结束	用户信息	
<ul> <li>○ 张三 1852010****</li> <li>● 服务号</li> <li>▲: 接入成功, 超級管理员为您服务</li> </ul>	NUNU		减后离码则则州广发这 计计则 胚间	administrator 2025-04-23 15:01:54 会话已结束	用户ID 用户昵称	m9tkv2j6-kutf-2n 张三 1852010****
		张三 1852010**** 2025-04-23 15:01:58 人工客服			米詞	די לל או

修改主包 pages/index/index.js ,设置 nickName 和 avatar 并将其传递给客服组件。

data: {
// SDKAppID/userID/userSig <b>的获取请参考步骤</b> 4
<pre>const SDKAppID = 0;</pre>
const userID = '';
<pre>const userSig = '';</pre>
const <b>nickName = '';</b>
const <b>avatar = '';</b> // <b>用户头像</b>
wx.navigateTo({
url: `/ai-desk-customer-wechat/pages/index/index?
SDKAppID=\${SDKAppID}&userID=\${userID}&userSig=\${userSig}`



+	`&nickName=\${nickName}`
+	`&avatar=\${avatar}`
});	
},	
})	

# 工具栏快捷按钮

如果您想实现输入框上方增加快捷按钮方便用户使用,例如增加"人工客服","查询包裹"等,效果如下所示:



修改主包 pages/index/index.js ,设置 toolbarButtonList 并将其转成 json 字符串后传递给客服组件。 toolbarButtonList 是一个包含了一个或多个配置的数组,配置的描述如下:

参数	类型	是否必填	说明
title	String	Yes	button 标题
icon	String	No	button 图标 url
isPreset	Number	Yes	<ul> <li>● 1 - 客服组件内置功能</li> <li>● 0 - 非内置功能</li> </ul>
presetId	String	No	当 isPreset 为 1 时,可选值如下: • humanService – 人工服务 • serviceRating – 服务评价 • endHumanService – 结束对话



isEnable d	Number	Yes	<ul> <li>1 - 渲染</li> <li>0 - 不這染</li> </ul>
content	String	No	当 isPreset 为 0 时,填入文本内容或者 url
type	Number	Yes	<ul> <li>1 - 点击 button 后客服组件发送 content 对应的文本</li> <li>2 - 点击 button 后客服组件打开 content 对应的 url(小程序打开 url 可能会受限,请提前参考相关文档解决)。</li> </ul>
<pre>Page({     data: {         },         getSuppo         // SDK         const         const</pre>	rt() { AppID/userID/n SDKAppID = 0; userID = ''; userSig = ''; toolbarButtonI tle":" <b>人工服务</b> " qcloud.com/too vice","isEnabi tle":" <b>服务评价</b> " myqcloud.com/f eRating","isEn tle":" <b>据束对话</b> " myqcloud.com/f anService","i: tle":" <b>智能客服</b> " igateTo({ `/ai-desk-cus {SDKAppID}&uso	<pre>List = [ /,"icon":"https blbar_button_1. led":1}, /,"icon":"https toolbar_button_ habled":1}, /,"icon":"https toolbar_button_ sEnabled":1}, /,"type":1,"con stomer-wechat/g erID=\${userID}&amp; onList=\${JSON.states}</pre>	参考步骤4 :://tccc-im-agent-avatar-1258344699.cos.ap- .png","type":1,"content":"","isPreset":1,"presetId" :://tccc-im-agent-avatar-sg-1258344699.cos.ap- _2.png","type":1,"content":"","isPreset":1,"presetI :://tccc-im-agent-avatar-sg-1258344699.cos.ap- _3.png","type":1,"content":"","isPreset":1,"presetI ttent":"智能客服高","isPreset":0,"isEnabled":1} pages/index/index? suserSig=\${userSig}` stringify(toolbarButtonList)}`

#### })

# 底部快捷订单

· 说明:

请升级 ai-desk-customer-wechat 到 v1.5.2 或更高版本。

如果您想实现打开客服会话时在聊天区域底部展示快捷订单,可以配置 bottomQuickOrder ,效果如下所示:





修改主包 pages/index/index.js , 设置 bottomQuickOrder 并将其转成 json 字符串后传递给客服组件。 bottomQuickOrder 的参数描述如下:

参数	类型	是否必填	说明
header	String	Yes	订单标题,对应上图的"智能客服高级版"
desc	String	Yes	订单描述,对应上图的"3000/月"
pic	String	No	订单图片 url
url	String	Yes	当客服坐席收到订单消息时,点击可打开此 url 对应的页面
customF ield	Array	No	自定义配置,如"订单号","订单时间"等信息。

```
Page({
    data: {
    },
    getSupport() {
        // SDKAppID/userID/userSig 的获取请参考步骤4
        const SDKAppID = 0;
        const userID = '';
        const userSig = '';
        const toolbarButtonList = [
```



```
{"title":"人工服务","icon":"https://tccc-im-agent-avatar-1258344699.cos.ap-
      {"title":"服务评价","icon":"https://tccc-im-agent-avatar-sg-1258344699.cos.ap-
      {"title":"结束对话","icon":"https://tccc-im-agent-avatar-sg-1258344699.cos.ap-
      {"type":1,"title":"智能客服高级版","content":"智能客服高级
版","isPreset":0,"isEnabled":1}
     header: "智能客服高级版",
     desc: "¥3000/月",
     customField: [
         name: '订单号',
         name: '下单时间',
     url: `/ai-desk-customer-wechat/pages/index/index?
SDKAppID=${SDKAppID}&userID=${userID}&userSig=${userSig}`
       + `&toolbarButtonList=${JSON.stringify(toolbarButtonList)}`
       + `&bottomQuickOrder=${JSON.stringify(bottomQuickOrder)}`
```

#### 多客服号

**① 说明:** 请升级 ai-desk-customer-wechat 到 v1.5.3 或更高版本。

如果您的应用内需要多个客服号,为用户提供专属客服功能,以提高服务质量和响应速度,可用 UIKit 提供的多客服号功能实现。效果如 下所示:



修改主包 pages/index/index.js ,设置 customerServiceIDList 和 customerServiceID 并将其转成 json 字符串后传 递给客服组件。

```
// 1. 初始化前将业务侧自定义的客服号 ID 传递给 UIKit
// 自定义客服号请使用 restapi - https://cloud.tencent.com/document/product/269/111798
const customerServiceIDList = ['customerServiceID_001', 'customerServiceID_002',
'customerServiceID_003'];
// 2. 初始化时,带上要发起会话的客服号 ID,比如 'customerServiceID_002'
// 如果不带上要发起会话的客服号 ID,则 UIKit 默认使用客服号 ID 列表的第 1 个元素
wx.navigateTo({
    url: `/ai-desk-customer-wechat/pages/index/index?
SDKAppID=${SDKAppID}&userID=${userID}&userSig=${userSig}`
    + `&customerServiceIDList=${JSON.stringify(customerServiceIDList)}`
    + `&customerServiceIDList=${JSON.stringify(customerServiceIDList)}`
    + `&customerServiceID=customerServiceID_002`
});
```

# 常见问题

# 什么是 UserSig? 如何生成 UserSig?

UserSig 是用户登录即时通信 IM 的密码,其本质是对 UserID 等信息加密后得到的密文。

UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端,并提供面向项目的接口,在需要 UserSig 时由您的项目向业务服务器 发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig 。

#### 小程序如果需要上线或者部署正式环境怎么办?

请在微信公众平台 > 开发 > 开发管理 > 开发设置 > 服务器域名中进行域名配置。域名配置详细参见:小程序合法域名。





# Flutter

最近更新时间: 2025-07-15 20:33:52

# 介绍

专为客服场景定制的 Customer UlKit,提供针对性强的用户侧客服会话界面,满足客服场景需求。UI、交互及功能体验,均面向智能 客服场景设计。

此外, Customer UIKit 让集成客服模块省去集成 IM,只需要简短的若干行代码,即可完成开发。

## 功能展示

15:49	::!! ? 98	15:51 🛛 🔹 扰 🗢	33	e) ::!! ♀ 97
🕻 😱 Hi, 我是服务号		< <u>र</u> Hi, 我是服务号	く 🔃 Hi, 我是服务号	
15:49			服务 很抱款给您带来了不好的体验 您的反馈能帮助我们不断改进 提供更好的服务	2, ±和
您好,有什么可以帮您?		等待中,请稍候	我们会认真倾听您的意见,并 力帮助您解决问题!	-辰
欢迎卡片 智能客服提供哪些功能 如何修改机器人知识库	换一换 C ,	接入成功,Runlin为您服务		
客服在哪里收发消息		8 你好,很高兴为你服务 👎 😋	: 查看內容	
如何设置工作时间		Eġ	再见 再见 感谢您的耐心和理解!	2
已读 智能客服	提供哪些功能	再见		已读 退货
您好,我们提供了开箱即用的智 能客服来应答用户咨询,包括智		请对本次服务进行评价	请选择您的购买渠道	
能机器人、客服工作台、多渠迫 消息聚合、路由分配、快捷回 复、满意度评价等功能。		请对本次服务进行评价	线下门店	
		非常满意	线上商城	
各 人工服务		各 人工服务	る 人工服务	
Ŷ	© +	Ų ©	+	☺ +

您可扫码安装 Demo app 体验使用效果。



# 前提条件

了解在线客服相关术语及相关配置,并已完成以下步骤:创建腾讯云 IM 应用、开通智能客服、登录客服管理端、获取客服号 ID,详情请 参见 快速入门。

# 环境与版本

腾讯云

- Flutter 版本: Flutter 3.24 Flutter 3.27。
- 支持模拟器调试及真机运行。

#### () 说明:

- 1. 如果您的项目 Flutter 版本较低, 建议升级至 Flutter 3.24 使用。如果确实不方便升级, 可使用 旧版本 Flutter IM UIKit + 客服插件方案。
- 2. 如果您使用 Flutter 3.29,请先降级至 Flutter 3.24 使用。使用 flutter downgrade 或 git 方式 checkout 管理。
- 3. 对于 Flutter 项目,我们不建议使用 Webview 集成智能客服网站渠道,因 Flutter Webview 针对拉起媒体、拍摄场景, 存在部分兼容性问题。因此,请尽量优先选用本 UIKit 方案。

### 快速集成

#### Demo 示例

建议您下载并参考下列步骤的 Demo 及其源码,配合阅读,以便更好的接入。

#### 步骤1: 集成包

本 UIKit pub package 包名为 tencentcloud\_ai\_desk\_customer。

flutter pub add tencentcloud\_ai\_desk\_customer

#### 步骤2: 权限配置

由于 Customer UIKit 运行,需要拍摄/相册/录音/网络等权限,需要您在 Native 层的文件中手动声明,才可正常使用相关能力。

#### Android

打开 android/app/src/main/AndroidManifest.xml ,在 <manifest></manifest> 中,添加如下权限。

```
<uses-permission
android:name="android.permission.INTERNET"/>
<uses-permission
android:name="android.permission.RECORD_AUDIO"/>
<uses-permission
android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission
android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission
android:name="android.permission.VIBRATE"/>
<uses-permission
android:name="android.permission.VIBRATE"/>
<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.CAMERA"/>
<uses-permission</td>
```



android:name="android.permission.READ\_MEDIA\_IMAGES"/>

uses-permission

android:name="android.permission.READ\_MEDIA\_VIDEO"/>

打开应用层的 build.gradle 文件,增加以下代码,以解决升级 Flutter 3.24 后,某些第三方库编译版本小于 31 导致的的编译报 错。放置位置可参见 Demo 示例代码 。



#### iOS

打开 ios/Podfile ,在文件末尾新增如下权限代码。并根据需要, 在 info.plist 文件中对应增加拍照、麦克风、相册等权限声明。



同时,还需要在 ios/Runner/Info.plist 文件中(或直接使用 Xcode 打开工程后编辑 Info.plist), 声明相关上述媒体权限的申 请提示, 您可按需修改如下描述.

<key>NSCameraUsageDescription</key>



<string>Tencent Customer UIKit needs access to your camera to take photos and</string>
videos.
<key>NSLocationWhenInUseUsageDescription</key>
<string>Tencent Customer UIKit needs access to your location to provide accurate</string>
customer service.
<key>NSMicrophoneUsageDescription</key>
<string>Tencent Customer UIKit needs access to your microphone to record audio.</string>
<key>NSPhotoLibraryAddUsageDescription</key>
<string>Tencent Customer UIKit needs permission to add photos to your photo</string>
library.
<key>NSPhotoLibraryUsageDescription</key>
<string>Tencent Customer UIKit needs access to your photo library to save photos</string>
and videos.
<key>NSUserTrackingUsageDescription</key>
<string>Tencent Customer UIKit uses tracking to improve your customer service</string>
experience.

# 步骤3: 登录与初始化

调用 init 方法完成 UIKit 初始化、登录及全局配置。 需要使用 IM 的初始化及登录参数, 具体可参见 即时通信 IM 登录鉴权 。

```
import 'package:tencentcloud_ai_desk_customer/tencentcloud_ai_desk_customer.dart';
TencentCloudAIDeskCustomer.init(
    sdkAppID: SDKAppID, // 腾讯云 IM 控制台创建应用的 SDKAppID
    userID: userID, // 详情可参见腾讯云 IM 登录鉴权
    userSig: userSig, // 详情可参见腾讯云 IM 登录鉴权
    config: TencentCloudCustomerConfig(), // 可选全局默认配置, 具体参数及使用方式可查看该类注释
);
```

• SDKAppID 信息,可在 即时通信 IM 控制台 单击**应用管理 > 创建新应用**,并选择**客服服务 Desk > 智能客服**,开通智能客服 后获取。

🔁 应用管理	智能客服	16 ng > 当前数据中心:中国 ① IM 技术服务交流群 I2 IM 出海交流专区
功能服务		
⑦ 消息服务 Chat ~		
☞ 推送服务 Push ~		开通智能客服
图 客服服务 Desk		开箱即用的智能客服系统、提供包括智能机器人、客服工作台、多渠道消息聚合、路由分配、满意度评价、数
• 智能客服		据看板等功能。查看效果展示。 🖸
▲ 音视频服务 RTC		✓ 我已阅读并同意《智能客服 Desk 计费说明》 Ľ
通用工具		
③ 回调配置		免费试用 立即开通 ー ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
名 UserSig生成校验		
三、 自助排障日志		



日 应用管理	<b>应用管理</b> IM 技术服务交流群	IM 出海交流专区						
功能服务	创建新应用	1.复制 5DKF	AppID					
⑦ 消息服务 Chat ~	应用名称	SDKAppID	应用状态	数据中心()	T	标签()	创建时间 ↓	服务信息
⑦ 推送服务 Push ∨		复制						消息服务 Chat 体验版 ▶
<ul> <li>智能客服</li> </ul>	智能客服现网使用	16 2 🗗	正常	中国		-	2025-01-02	推送服务 Push 试用版 ▶
Ⅰ 音视频服务 RTC								智能客服 Desk 高级版 ▶

- userID 信息,可本地生成一个随机的字符串,例如 test-1234。
- userSig 信息,可单击 即时通信 IM 控制台 > UserSig生成校验,填写创建的 userID,即可生成 userSig。

🕞 应用管理	UserSig生成&校验 16 现 > 当前数据中心:中国① IM 技术服务交流群 2 IM 出	海交流专区
功能服务		
⑦ 消息服务 Chat ~	签名(UserSig)生成工具	登录鉴权介绍 🖸
☞ 推送服务 Push ~	此工具可以快速生成签名(UserSig),用于本地跑通 Demo 以及功能调试。	
图 客服服务 Desk	用户名(UserID) test-1234 •	
国 音视频服务 RTC	密钥	
通用工具	3.填入创建的 userID	
☆ 回调配置		
2 UserSig生成校验		
☴ 自助排障日志		
	如有需要,请到应用基础信息中复制密钥	
2.UserSig 生成	生成签名(UserSig) 校验	
	4.生成签名并复制	
	复制签名 (UserSig)	

#### 步骤4: 打开客服聊天页

调用 navigate 方法,跳转至客服聊天页面。

<pre>import 'package:tencentcloud_ai_desk_customer/tencentcloud_ai_desk_customer.dart';</pre>
TencentCloudAIDeskCustomer.navigate(
customerServiceID: "@customer_service_account", // 可选, 如不配置多客服号或修改默认的客服号
UserID, 可无需传入. 各服号 UserID 可从智能各服官埋ss鱼看: https://desk.qcloud.com/ config: TencentCloudCustomerConfig(), // 可选针对该客服会话的特殊配置, 具体参数及使用方式可查
看该类注释.此处仅需手动指定相比全局配置的增项和修改项即可,其余配置来自全局默认配置. );

至此,智能客服功能在 Flutter 端集成完成。

# 更多功能

多语言



如果您的 Desk 套餐包支持多语言,UIKit 可自动跟随系统语言或使用您指定的语言。

TencentCloudCustomerConfig **类中有一个可选配置项** language 用于指定需要的语言。既可 init 时全局配置,也可 navigate 时指定。

如果不指定,默认采用系统语言。若系统语言不支持,兜底使用简体中文(国内站)、英文(国际站)。具体语言支持如下。

#### () 版本说明:

本功能仅在 tencentcloud\_ai\_desk\_customer: ^1.2.0 及后续版本提供。版本语言支持情况如下:

• 1.2.0: 英语、简体中文、繁体中文、日语、印尼语。

<pre>import 'package:tencentcloud_ai_desk_customer/tencentcloud_ai_desk_customer.dart';</pre>
<pre>import 'package:tencent_desk_i18n_tool/language_json/strings.g.dart';</pre>
TencentCloudAIDeskCustomer.init(
········// 其他配置项
<pre>config: TencentCloudCustomerConfig(</pre>
<b>language:</b> TDeskAppLocale.en, // <b>如果不手动指定,默认使用系统语言</b>
·······// 其他配置项
<pre>config: TencentCloudCustomerConfig(</pre>
language: TDeskAppLocale.en, // 如果不手动指定,默认全局配置或使用系统语言

#### 语言枚举值如下:



#### 注销或切换用户

如果您需要注销当前登录用户,或切换登录用户,请先 dispose 当前实例,具体代码示例如下:



import 'package:tencentcloud\_ai\_desk\_customer/tencentcloud\_ai\_desk\_customer.dart';

#### TencentCloudAIDeskCustomer.dispose();

#### 用户端带昵称和头像登录

#### () 版本说明:

本功能仅在 tencentcloud\_ai\_desk\_customer: ^1.5.0 及后续版本提供。

如果人工客服在工作台接待用户咨询时,希望能看到用户的昵称、头像等信息以提升沟通效率,效果如下图所示:

我	的	$\oplus$	张三 1852010**** @服务号		ⓒ 转接 🕛 结束	用户信息	
9	张三 1852010**** @服务号 各:接入成功,超级管理员为您服务	刚刚		ぶつ成ぶ川のHF 友达 J 叶川 巡 H	administrator 2025-04-23 15:01:54 会话已结束	用户ID 用户昵称	m9tkv2j6-kutf-2n 张三 1852010****
			张三 1852010**** 2025-04-23 15:01:58 人工客服			来源	服务号

#### 请在初始化时,补充传入昵称和头像即可。

```
import 'package:tencentcloud_ai_desk_customer/tencentcloud_ai_desk_customer.dart';
TencentCloudAIDeskCustomer.init(
    nickName: '',
    avatar: '',
    // ... 其他配置, 可见 步骤3: 登录与初始化
);
```

# 自定义样式及能力

如快速集成部分的基础配置(TencentCloudCustomerConfig)无法满足您业务的自定义需求,您可参考本部分,本地引入源码,进 行修改,后应用于项目中。

## Fork 并 Clone 代码

- 1. 访问本工具包的 GitHub 仓库: tencentcloud\_ai\_desk\_customer\_flutter。
- 2. 单击仓库页面右上角的 "Fork" 按钮。这将在您的 GitHub 账户下创建一个仓库副本。
- 3. 将 fork 的仓库克隆到本地机器上的一个目录中。您可以使用以下 Git 命令执行此操作:

git clone https://github.com/<your-username>/tencentcloud\_ai\_desk\_customer\_flutter.git

将 <your-username> 替换为您的 GitHub 用户名。

4. 在您项目的 pubspec.yaml 文件中,将 fork 的仓库的本地路径添加到 dependencies 部分,并替代原线上版本依赖:

lependencies: tencentcloud\_ai\_desk\_customer\_flutter: path: /path/to/your/local/repository

将 /path/to/your/local/repository 替换为本地机器上克隆的仓库的实际路径。 您可在当前 IDE 环境、也可新开一个 IDE 窗口,打开本地引入的 UIKit 源码目录文件夹,并进行修改。



以下是一些场景说明,您可不局限于下方说明,针对整个项目,所有代码都能进行修改。

#### 修改图片资源

图片资源,可直接找到现有图片,替换为同名文件即可。

• 针对客服场景(如聊天背景、评分星星)的图片资源位于 lib/customer\_service/assets 目录。

• 聊天通用(如菜单中icon)的图片资源位于images 目录。

#### 修改聊天基础配置及操控

客服 Customer UIKit 的聊天模块,命名为 TencentCloudCustomerMessage 。其大体基于 IM UIKit 的 Chat 组件,因此 IM 的 众多基础配置,均可直接复用并修改。

因此, TencentCloudCustomerMessage 即为基于 IM UIKit Chat 组件 TIMUIKitChat 的客服场景适配版,用法基本一致。 您可在 lib/customer\_service/widgets/tencent\_cloud\_customer\_message\_container.dart 文件中找到
TencentCloudCustomerMessage 的实现。

修改此文件,进行**聊天能力基础配置** config: TIMUIKitChatConfig、toolTipsConfig: ToolTipsConfig、morePanelConfig: MorePanelConfig,**聊天能力生命周期管理** lifeCycle: ChatLifeCycle,**各类 UI 组件 Builder** 如 inputTopBuilder tongueItemBuilder customAppBar messageItemBuilder。除此配置外,还可使用 \_chatController 变量存储的 **ChatController,控制该 Chat 组件**。

以上配置、Builder 及 Controller 的用法,均和 IM UIKit TIMUIKitChat 组件中同名参数的使用方式一致,您可参考对应注释,了 解其用法;或参考下方内容,了解其与智能客服场景结合的案例。

#### 客服消息自定义样式及渲染

智能客服的特殊消息,例如<mark>分支消息、评价消息、卡片消息</mark>等,都是通过自定义消息实现,因此您需要在接收消息时判断消息类型并渲 染。 <mark>isCustomerServiceMessage</mark> 方法用于判断自定义消息是否属于智能客服的自定义消息。使用方法如下:

在线客服消息中有一些特殊的消息为标志消息,例如**会话结束、会话开始、客服配置**等,这一类消息不需要渲染在消息列表中。 <mark>isCustomerServiceMessageInvisible</mark> 方法用于判断在线客服消息是否需要在消息列表中渲染。使用方法如下:

```
// 使用 TencentCloudCustomerMessage 组件的 lifeCycle 属性实现
TencentCloudCustomerMessage(
    lifeCycle: ChatLifeCycle(messageShouldMount: (V2TimMessage message) {
        if (TencentCloudChatCustomerServicePlugin
            .isCustomerServiceMessageInvisible(message) &&
        TencentCloudChatCustomerServicePlugin.isCustomerServiceMessage(
            message)) {
            return false;
            }
            return true;
        }),
    }
}
```



#### );

在线客服消息中,评价消息因为不需要会话气泡且需要独占一行,因此, 此消息需要单独处理。 tencent\_cloud\_chat\_customer\_service\_plugin 提供了**isRowCustomerServiceMessage** 方法用于判断消息是否需要独 占一行。使用方法如下:



## 主动评价客服

isCanSendEvaluate 方法用于快速判断是否能主动发送评价,isCanSendEvaluateMessage 方法快速判断是否为发送评价配置 消息,同时也提供了 getEvaluateMessage 方法拉取用户评价。使用方法如下:



```
title: TIM_t("服务评价"),
showName: TIM_t("服务评价"),
 // 点击拉取评价
 TencentCloudChatCustomerServicePlugin.getEvaluateMessage(
```

# 主动结束人工会话

当用户端进入 人工会话状态 时,可以发送一条约定好类型的自定义消息,发送后服务端会主动断开此次人工会话服务。 isInSession 方法用于快速判断是否在人工会话中,isInSessionMessage 方法快速判断是否为标识人工会话状态消息,同时也提供 了 sendCustomerServiceEndSessionMessage 方法快速发送结束会话消息。使用方法如下:

```
// TencentCloudCustomerMessage 组件的 morePanelConfig、additionalDesktopControlBarItems、
newMessageWillMount 属性实现
TencentCloudCustomerMessage(
lifeCycle: ChatLifeCycle(
```



```
TencentCloudChatCustomerServicePlugin.isInSession(message) &&
    Icons.local_phone_outlined,
title: TIM_t("结束会话"),
```





完成修改源码后,可将变更提交至您 fork 的仓库。

# 同步更新本 UIKit 后续版本

Customer UIKit 的 GitHub 仓库中,每个版本,均通过 Tag 标记。您可同步最新代码到指定版本或位于 main 分支的最新版本。

☐ C RoleWong / tencentcloud_ai_desk_customer_flutter <> Code ⊙ Issues ↑↑ Pull requests ⊙ Actions ⊞ Projects □□ Wiki ⊙ Secu	rity 🗠 Insights 🕸 Settings				Q Type [] to
	tencentcloud_ai_desk_customer	_flutter Public	🔊 Pin 💿 Unwatch	1 ▼ <sup>92</sup> Fork 0 ▼ ☆ Star 0 ▼	
	🛇 main 👻 🐉 1 Branch 🛇 1 Tag	Q Go to file	t Add file - Code -	About	ŵ
	Switch branches/tags ×		Seab4bc - 11 minutes ago 🕚 1 Commit	Al-driven customer service UlKit for Tencent Cloud Desk (customer-side).	
	Q Find a tag	feat: 1.2.0	11 minutes ago	C Readme	
	Branches Tags	feat: 1.2.0 11 minutes ago	Ac Activity		
	1.2.0	feat: 1.2.0	11 minutes ago	☆ 0 stars	
	View all tags	feat: 1.2.0	11 minutes ago	1 watching     0 forks	
	🕒 .gitignore	feat: 1.2.0	11 minutes ago		
	🗅 .metadata	feat: 1.2.0	11 minutes ago	Releases	

如需将您 fork 的仓库同步官方仓库,可使用 pull upstream 操作。

以下是一个 pull upstream 的 Git 操作示例:

1. 首先,在您的本地仓库中添加上游(upstream)远程仓库:

git remote add upstream
https://github.com/RoleWong/tencentcloud\_ai\_desk\_customer\_flutter.git

2. 拉取上游仓库的最新更改:

git fetch upstream

3. 将您的本地仓库切换到要更新的分支(例如 main 或 master ):

git checkout main

4. 将上游仓库的更改合并到您的本地仓库:

git merge upstream/main

- 5. 如果有冲突,请在编辑器中解决它们,确保各个 JSON 词条库是完整的。
- 6. 提交解决冲突后的更改:

git add



t **commit -m "**Merge upstream changes and resolve conflicts'

7. 将更改推送到您的远程仓库:

git push origin main

现在,您的 fork 版本已经包含了线上版本的对应代码。



# iOS

最近更新时间: 2025-07-14 17:10:52

# 介绍

专为客服场景定制的 Customer UlKit,提供针对性强的用户侧客服会话界面,满足客服场景需求。UI、交互及功能体验,均面向智能 客服场景设计。

此外, Customer UIKit 让集成客服模块省去集成 IM,只需要简短的若干行代码,即可完成开发。

#### 前提条件

了解在线客服相关术语及相关配置,并已完成以下步骤:创建腾讯云 IM 应用、开通智能客服、登录客服管理端、获取客服号 ID,详情请 参见 快速入门。

# 功能展示

🥶 Hi, 我是智能客服		く 😶 Hi, 我是智能客服		く 🥶 Hi, 我是智能客服	l
您好,有什么可以帮您?			人工服务	您好,有什么可以帮您?	
次迎卡片	ð	等待中,请稍候		欢迎卡片	C
智能客服提供哪些功能		: 接入成功,Runlin为您服务		智能客服提供	哪些功能
客服在哪里收发消息				客服在哪里收	<b>文发消息</b>
机器人功能可以关闭吗		您好,请问有什么可以帮到您		机器人功能可	以关闭吗
如何设置工作时间			都解决啦,谢谢	如何设置工	作时间
智能客服提供	+哪些功能	好的那这边先不打扰您啦			功能种类
		感谢您使用我们的服务,请对此	次服务进行评价!		
您好,我们提供了开箱即用的智 能客服来应答用户咨询,包括智		请对本次服务进行.		您可能想问:	
能机器人、客服工作台、多渠迫 消息聚合、路由分配、快捷回 5. 进音座迎公等功能		* * * *	* *	智能客服提供	哪些功能
复、 <i>两总</i> 反叶川寺功能。		提交评价		支持哪些消	息渠道
	人工服务			支持哪些用户	咨询渠道?
等待中,请稍候		会话已结束		怎么实现所有的问题	<b>夏都由客服回复</b>
人工服务		8 人工服务		8 人工服务	

# 快速集成

# 环境与版本

- iOS 版本: iOS 13 以上。
- 支持模拟器调试及真机运行。

# Demo 示例

建议您下载并参考下列步骤的 demo 及其源码,配合阅读,以便更好的接入。

# CocoaPods 集成



#### 方式1: 拉取远程 CocoaPods 集成

在 XCode 工程中, 通过 Cocoapods 集成 TencentCloudAlDeskCustomer。 在 Podfile 中, 添加如下示例代码:



#### 方式2: DevelopPods 源码集成(有源码修改时推荐)

从 GitHub 下载 TencentCloudAlDeskCustomer-iOS 、TDeskCustomerServicePlugin、TDeskChat、 TDeskCommon、TDeskCore 。直接拖入您的工程目录下: TDesk。



## 登录与初始化

调用 loginWithSdkAppID 方法完成 UIKit 登录,并调用 setCustomerServiceUserID 设置在线客服的 UserID,示例代码如 下:

<pre>#import "TencentCloudAIDeskCustomer/TencentCloudCustomerManager.h"</pre>
- (void)login:(NSString *)userID userSig:(NSString *)sig {
[[TencentCloudCustomerManager sharedManager] loginWithSdkAppID:"应用的    SDKAppID"
userID: <b>"当前登录用户的</b> UserID <b>"</b> userSig: <b>"当前登录用户的</b> UserSig <b>"</b> completion:^(NSError *error) {
if (error.code == 0) {
// <b>如果没有修改默认的客服号 ID 就不需要调用该函数。可从智能客服管理端首页查看</b> :
// [[TencentCloudCustomerManager sharedManager] setCustomerServiceUserID:
"@customer_service_account"];
} else {
NSLog(@ <b>"登录失败"</b> );
}];

 SDKAppID 信息,可在 即时通信 IM 控制台 单击应用管理 > 创建新应用,并选择客服服务 Desk > 智能客服,开通智能客服 后获 取。



功能服务         ③ 消息服务 Chat ~         ⑦ 推送服务 Push ~         ◎ 客服服务 Desk ~         • 智能客服         ● 音视频服务 RTC         適用工具
・ 消息服务 Chat ◇           ・ 行意服务 Chat ◇             ・ 推送服务 Push ◇         ・         ・           ・             ・           客服服务 Desk ◇         ・             ・               ・               ・               ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・           ・             ・         ・
<ul> <li>ア 推送服务 Push ◇</li> <li>● 客服服务 Desk ◇</li> <li>・ 智能客服</li> <li>□ 音视频服务 RTC</li> <li>通用工具</li> </ul>
<ul> <li>○ 客服服务 Desk へ</li> <li>・ 智能客服</li> <li>一 音视频服务 RTC</li> <li>通用工具</li> </ul>
□ 音视频服务 RTC
各弗泽巴 古明正语 <b>一</b> 丁 <b>日八旦</b> 清
₽ UserSig生成校验
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

<b>己</b> 应用管理	应用管理 IM 技术服务交流群	IM 出海交流专区					
功能服务	创建新应用	1.复制 5DKF	appID				
⑦ 消息服务 Chat ~							
☞ 推送服务 Push ·	应用名称	SDKAppID	应用状态	数据中心(i)	☞ 标签()	创建时间 ↓	服务信息
客服服务 Desk ^		复制					消息服务 Chat 体验版 ▶
• 智能客服	智能客服现网使用	16 2 🖻	正常	中国	-	2025-01-02	推送服务 Push 试用版 ▶
音视频服务 RTC							智能客服 Desk 高级版 ►

● userID 信息,可本地生成一个随机的字符串,例如 test-1234。

• userSig 信息,可单击 即时通信 IM 控制台 > UserSig生成校验,填写创建的 userID,即可生成 userSig。

🕞 应用管理	UserSig生成&校验         16         現 >         当前数据中心:中国①         IM 技术服务交流群 12	IM 出海交流专区
功能服务		
⑦ 消息服务 Chat ~	签名(UserSig)生成工具	登录鉴权介绍 🖸
☞ 推送服务 Push ~	此工具可以快速生成签名(UserSig),用于本地跑通 Demo 以及功能调试。	
客服服务 Desk      ✓	用户名(UseriD) test-1234 •	
■ 音视频服务 RTC	密钥	
通用工具	3.填入创建的 userID ************************************	
③ 回调配置		
<b>2</b> UserSig生成校验		
☴ 自助排障日志		
	如有需要,请到应用基础信息中复制密钥	
	牛 <b>府签</b> 条(IlserSin)	
2.UserSig 生成	校验	
	4.生成签名并复制	
	复制签名(UserSig)	

# 打开客服聊天页

调用 pushToCustomerServiceViewControllerFromController 方法,跳转至客服聊天页面。

// 一定要确保已经登录成功了,否则直接跳转过去就是空白



[[TencentCloudCustomerManager sharedManager] pushToCustomerServiceViewControllerFromController:self];

至此,在线客服功能在 iOS 端集成完成。

## 高级用法

如果需要更多高级能力,可参考使用以下高级 API 能力。

#### 设置快捷用语

设置输入框上部快捷用语。

[[TencentCloudCustomerManager sharedManager] setQuickMessages:<#
(NSArray<TUICustomerServicePluginMenuCellData \*> \*) #>];



# Android

最近更新时间: 2025-07-14 17:10:52

# 介绍

专为客服场景定制的 Customer UlKit,提供针对性强的用户侧客服会话界面,满足客服场景需求。UI、交互及功能体验,均面向智能 客服场景设计。

此外,Customer UIKit 让集成客服模块省去集成 IM,只需要简短的若干行代码,即可完成开发。

# 前提条件

了解在线客服相关术语及相关配置,并已完成以下步骤:创建腾讯云 IM 应用、开通智能客服、登录客服管理端、获取客服号 ID,详情请 参见 快速入门。

# 环境与版本

- Android Studio
- Gradle-7.4.1
- Android Gradle Plugin Version-7.0.1

# 功能展示



# 快速集成



在项目的 setting.gradle 的 dependencyResolutionManagement 中加入以下内容:

repositories ·





# 步骤2:引入 AI Desk 包

我们提供了两种方式来引入,如不需要修改源码可直接使用 Maven 镜像的方式引入,如果需要修改源码可用源码集成的方式。



#### 源码方式集成

- 1. 下载 AI TDesk SDK 源码,把 aideskcustomer、deskchat、deskcommon、deskcontact、deskcore、 deskcustomerserviceplugin 文件夹复制到你项目的根目录。
- 2. 修改在项目的 settings.gradle 文件,引入 AI TDesk SDK 相关代码。

gin

3. 找到 app 的 build.gradle 文件, 然后在 dependencies 中添加 AI TDesk SDK 的依赖。

implementation project(':aideskcustomer')

# 步骤3: 用户登录

TencentAiDeskCustomerLoginConfig **config** = new TencentAiDeskCustomerLoginConfig(); // config**可选填** 



TencentAiDeskCustomer.getInstance().login(context, sdkAppID, userID, userSign, config,						
@Override						
System.out.println("login success");						
@Override						
public void onError(int code, String desc) {						
<pre>System.out.println("login failed"+code+","+desc);</pre>						

# 步骤4: 打开新的客服聊天页

startActivity(TencentAiDeskCustomer.getInstance().getTencentCloudCustomerChatIntent(cont
ext));

### 高级用法

#### 设置主题

TencentAiDeskCustomer.setTheme(TencentAiDeskCustomerThemeConfig.DARK);

#### 设置显示人工服务按钮

	设置为	true	后,	只有没再人工服务状态就会显示,在人工服务后会自动隐藏。		
	设置为	true	后,	永远不显示人工服务状态按钮		
<pre>TencentAiDeskCustomer.getInstance().setShowHumanService(true);</pre>						

#### 设置快捷用语

LinkedList<TencentAiDeskCustomerQuickMessageInfo> quickMessages = new LinkedList<TencentAiDeskCustomerQuickMessageInfo>();// 详情见快捷用语类 TencentAiDeskCustomer.getInstance().setQuickMessages(quickMessages);

#### 设置携带商品信息

TencentAiDeskCustomerProductInfo **info** = new TencentAiDeskCustomerProductInfo(); // **详情 见商品信息类** TencentAiDeskCustomer.getInstance().setProductInfo(**info**);

## 设置语言

如果您的 Desk 套餐支持多语言,UIKit 可以自动适配系统语言或使用您指定的语言。目前默认支持中文、英文和阿拉伯语。如需其他语 言,请联系我们。

如果未指定语言,系统将默认使用系统语言。如果系统语言不受支持,则默认使用简体中文(适用于国内站)。具体支持的语言如下。



// **设置为英文,**TencentAiDeskCustomerLanguageConfig.zh**为中文。** TencentAiDeskCustomer.getInstance().setLanguage(v.getContext

# 其他

如果想要快速跑通 Demo,并想修改部分效果,可下载 DeskDemo以及源码。