云联络中心

SDK 开发指南

产品文档

腾讯云

文档目录

SDK 开发指南

入门概述

最近更新时间：2024-04-01 17:17:05

云联络中心（Cloud Contact Center）帮助企业快速搭建集电话、在线交流、音视频通话为一体的客户联络平台。腾讯云联络中心提供客户联络的全套 SDK，通过阅读本文，您可以了解根据 SDK 实现客户联络的流程。

入门流程

您可以按照以下步骤进行开发集成：

步骤	操作
1	创建云联络中心应用
2	参考需要使用的客服类型，进行配置： 快速配置电话呼出 快速配置电话呼入
3	参考文档 集成座席端SDK 将座席端集成进您自己的系统
4	根据需要使用的客服类型，参考对应文档集成： （从侧边栏点导航浏览更方便） 集成电话客服

交流与反馈

点此进入 云联络中心 社群，享有专业工程师的支持，解决您的难题。

集成座席端 SDK
使用 Demo 快速运行
Web

最近更新时间：2024-04-01 17:22:00

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 云联络中心 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：2024-04-01 17:29:48

本文主要介绍如何快速跑通云联络中心 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。
您已 开通云联络中心 服务，并创建了 云联络中心实例 。
您已完成自有号码接入。并且完成了对应的 IVR 配置。

关键概念

SdkAppId：是用户在 云联络中心控制台 上创建的应用 ID，称之为 SdkAppId，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 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。

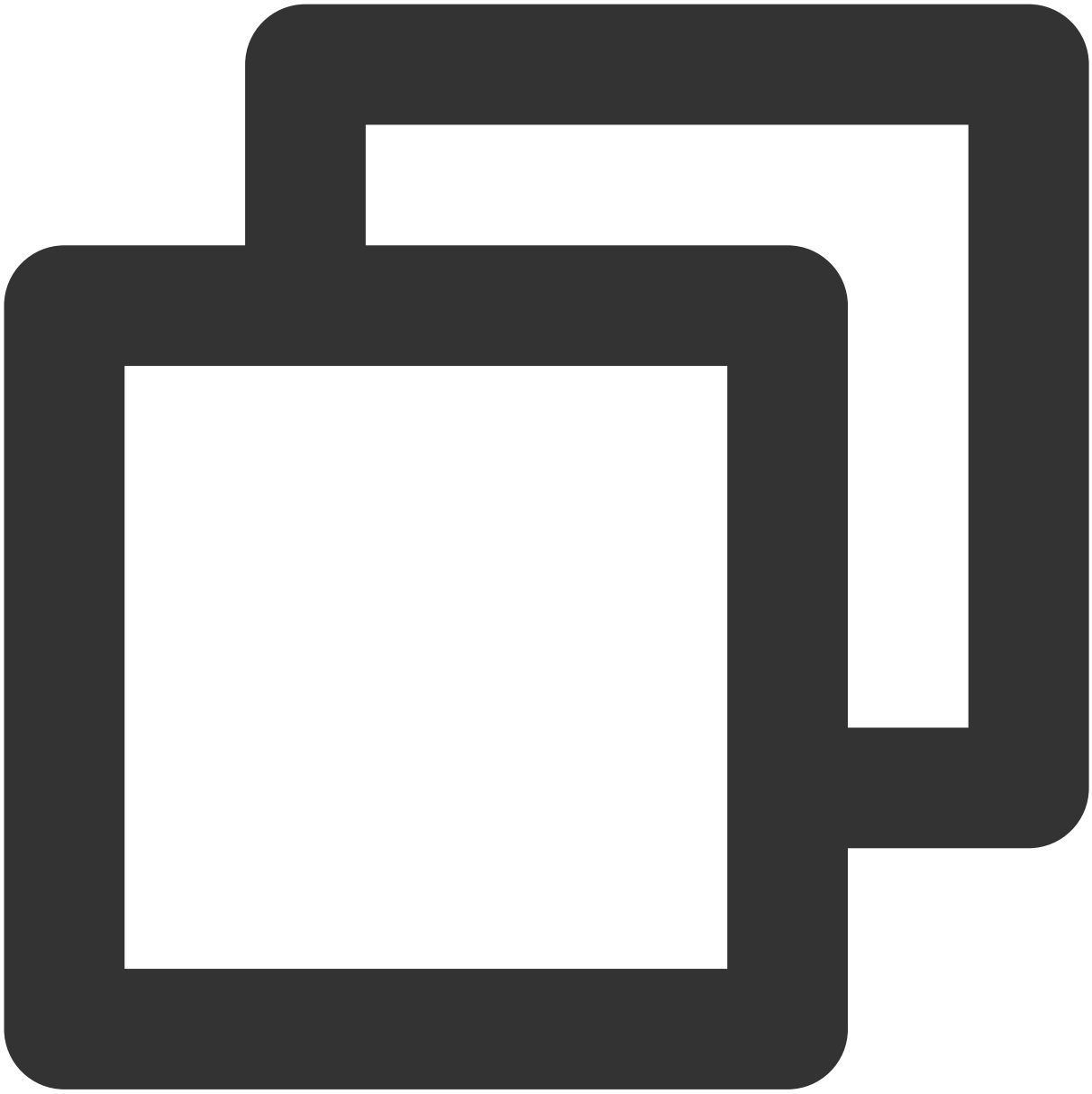
操作步骤

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**。

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。

genTestToken.js

```
/**
 * 座席账号，格式为 ： xxx@qq.com
 */
const USERID = 'xxx@qq.com';

/**
 * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
 * 进入腾讯云呼叫中心[控制台](https://console.cloud.tencent.com
 * 它是腾讯云用于区分客户的唯一标识。
 */
const SDKAPPID = 1400000000;
/**
* 计算签名用的加密密钥ID，[查看秘钥](https://console.cloud.tence
* 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥
* 文档：https://cloud.tencent.com/document/product/679/58260
*/
const SECRETID = "";

/**
* 计算签名用的加密密钥Key，[查看秘钥](https://console.cloud.tenc
* 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥
* 文档：https://cloud.tencent.com/document/product/679/58260
*/
const SECRETKEY = "";
```

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。

注意：

什么是自定义调试基座及使用说明,请参见 官方教程。

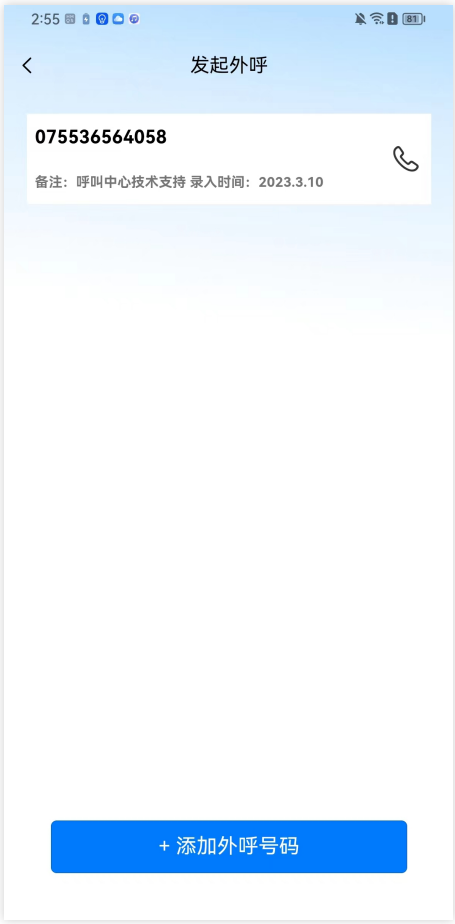
步骤5：运行

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



交流与反馈

点此进入 云联络中心 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：2024-04-01 17:31:18

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
您已 开通云联络中心 服务，并创建了 云联络中心实例 。
您已完成自有号码接入。并且完成了对应的 IVR 配置。

关键概念

SdkAppId：是用户在 腾讯云联络中心控制台 上创建的应用 ID，称之为 SdkAppId，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

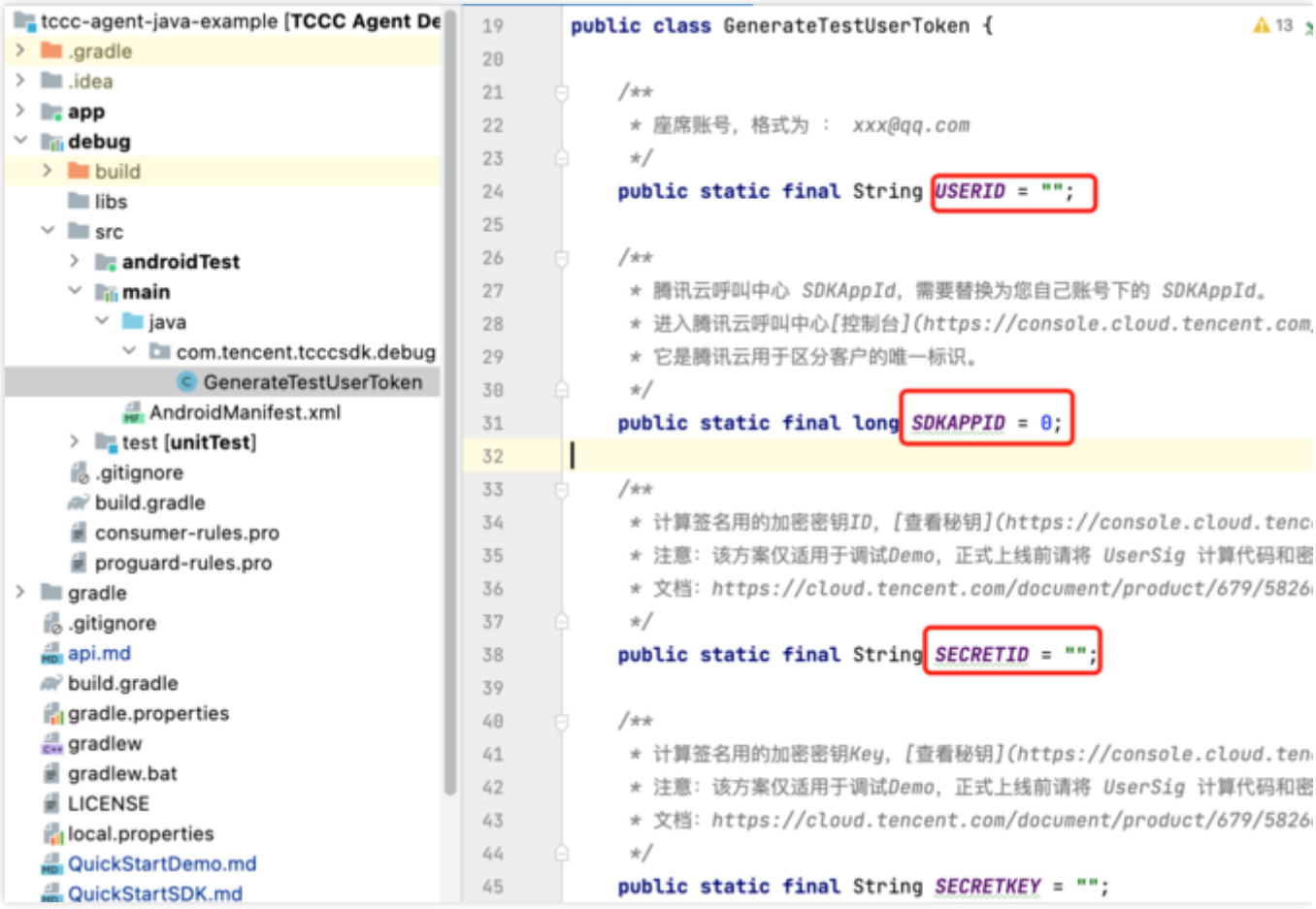
1. 找到并打开 `debug/src/main/java/com/tencent/tcccsdk/debug/GenerateTestUserToken.java` 文件。
2. 设置 GenerateTestUserToken.java 文件中的相关参数：

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。



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

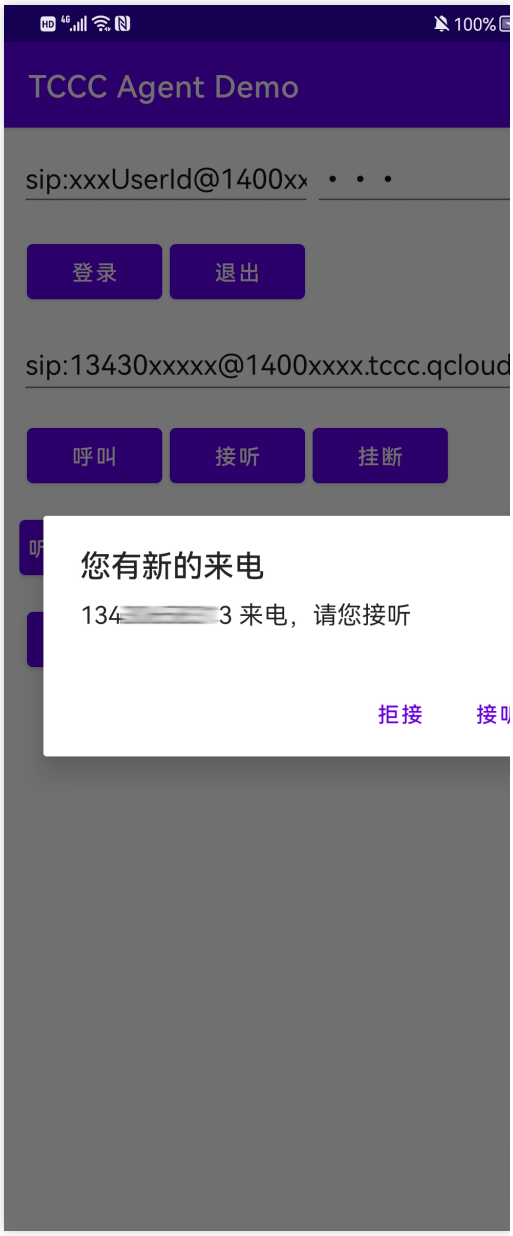
使用 Android Studio（3.5及以上的版本）打开源码工程 `tccc-agent-java-example`，单击**运行**即可。

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：2024-04-01 17:33:16

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 。
您已 开通云联络中心 服务，并创建了 云联络中心实例 。
您已完成自有号码接入。并且完成了对应的 IVR 配置。

关键概念

1. **SdkAppId**：是用户在 腾讯云联络中心控制台 上创建的应用 ID，称之为 SdkAppId，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

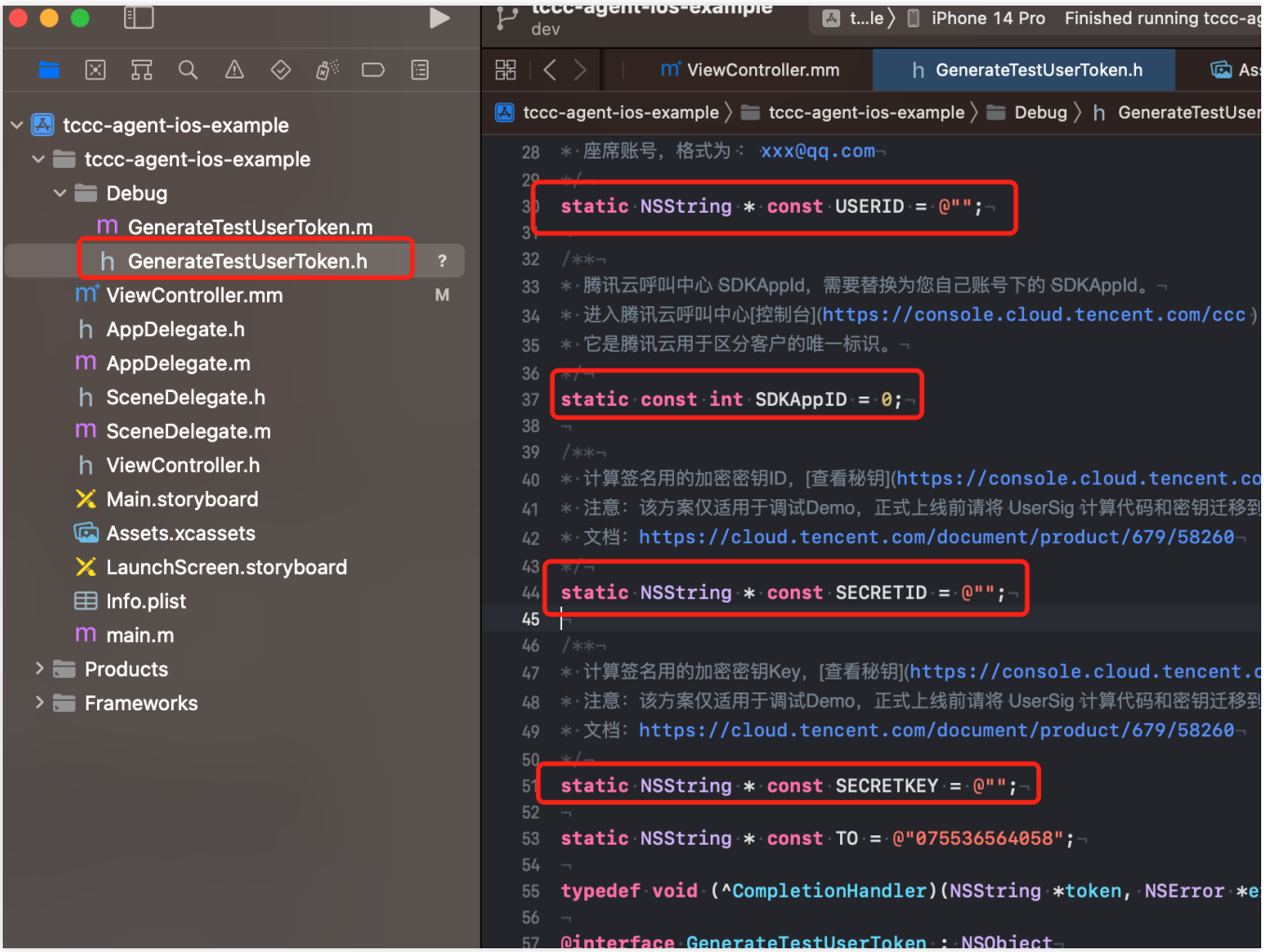
1. 找到并打开 debug/GenerateTestUserToken.h 文件。
2. 设置 GenerateTestUserToken.h 文件中的相关参数：

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

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。

座席端 SDK API 文档

Web

最近更新时间：2024-04-01 18:03:15

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<CommonSDKResponse>

参数		类型	必填	备注
options	phoneNumber	String	是	被叫号码
	phoneDesc	String	否	号码备注，在通话条中会替代号码显示
	uui	String	否	用户自定义数据，传入后可通过 电话 CDR 数据推送 推送返回
	skillGroupId	String	否	指定技能组内绑定的外呼号码
	callerPhoneNumber	String	否	指定外呼号码
	servingNumberGroupIds	String[]	否	指定号码 ID 列表
	phoneEncodeType	'number'	否	目前仅支持'number'，在开启 号码映射 时强制使用真实号码

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

CallResponse 描述如下：

参数		类型	必填	备注
response	sessionId	String	是	指定会话 ID
	calleeLocation	String	否	被叫号码归属地址
	calleePhoneNumber	String	是	被叫号码
	callerPhoneNumber	String	是	外呼时使用的主叫号码
	serverType	String	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 会话服务类型
	remark	String	否	被叫号码备注

接听会话

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
	userId	String	否	转接到指定座席

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

取消通话保持

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

参数		类型	必填	备注

options	sessionId	String	是	指定会话 ID

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

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
	userId	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
	userId	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 arrange: 话后整理 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、Edge80以上的浏览器。

返回麦克风设备列表

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

返回扬声器设备列表

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

UI（用户界面相关接口函数）

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

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

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

参数		类型	必填	备注
options	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
	calleePhoneNumber	String	是	被叫号码
	callerPhoneNumber	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
	protectedCallee	String	否	在开启号码映射时存在，表示被叫
	protectedCaller	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

参数		类型	必填	备注
options	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
	remark	String	否	备注
	peerSource	String	否	渠道来源
	channelName	String	否	自定义参数
	clientData	String	否	用户自定义参数

音频会话呼入

参数		类型	必填	备注
options	sessionId	String	是	会话 ID
	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
	calleeRemark	String	否	渠道入口备注

	userId	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
	peerSource	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
	clientData	String	否	用户自定义参数

视频会话呼入

参数		类型	必填	备注
options	sessionId	String	是	会话 ID
	type	'video'	是	视频会话类型
	timeout	String	是	会话接入超时时长，0代表不超时
	userId	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

内部会话呼入

参数		类型	必填	备注
options	sessionId	String	是	会话 ID
	type	'internal'	是	内部会话类型
	timeout	Number	是	会话接入超时时长，0代表不超时
	peerUserId	String	是	主叫座席的账号

座席接入会话

tccc.events.userAccessed

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

会话超时转接事件

tccc.events.autoTransfer

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

会话结束事件

tccc.events.sessionEnded

参数		类型	必填	备注
options	sessionId	String	是	指定会话 ID
	closeBy	String	是	表示挂断方： client：用户挂断 seat：座席挂断 admin：系统挂断 timer：定时器挂断
	mainReason	String	否	仅在电话类型，并且挂断方为"admin"时存在，表示挂断原因
	subReason	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

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

外呼对方接听事件

tccc.events.calloutAccepted

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

会话转接事件

tccc.events.transfer

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

座席状态变更事件

tccc.events.statusChanged

参数		类型	必填	备注	
options	status	AgentStatus	否	详细说明请参见 座席状态	

语音识别事件

tccc.events.asr

参数		类型	必填	备注	
options	sessionId	String	是	指定会话 ID	
	result	ASR识别结果	是	语音识别结果结构体	
	flow	'IN' | 'OUT'	是	识别方向 IN: 用户侧 OUT: 座席侧	

uni-app

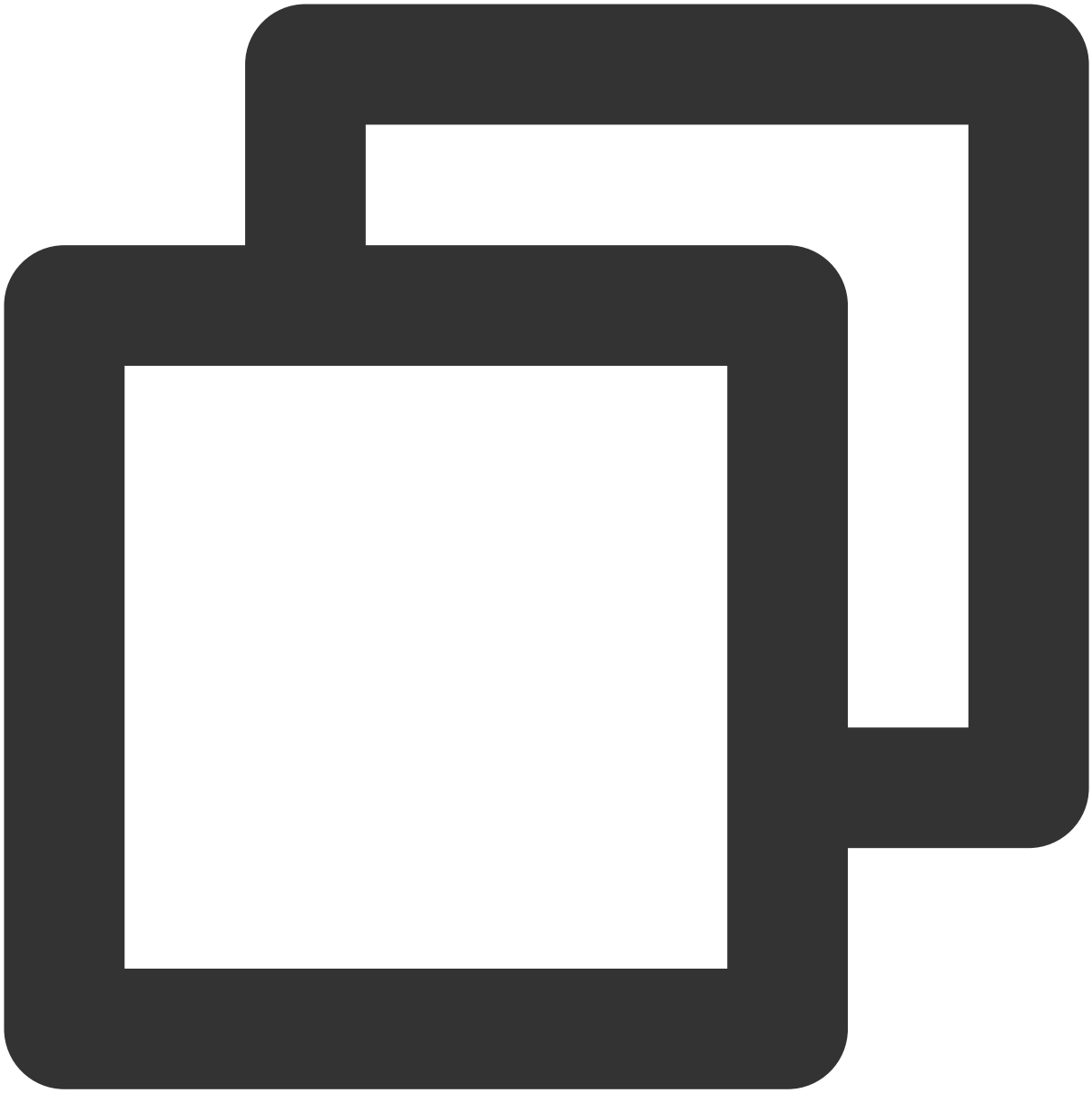
最近更新时间：2024-04-01 17:50:30

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



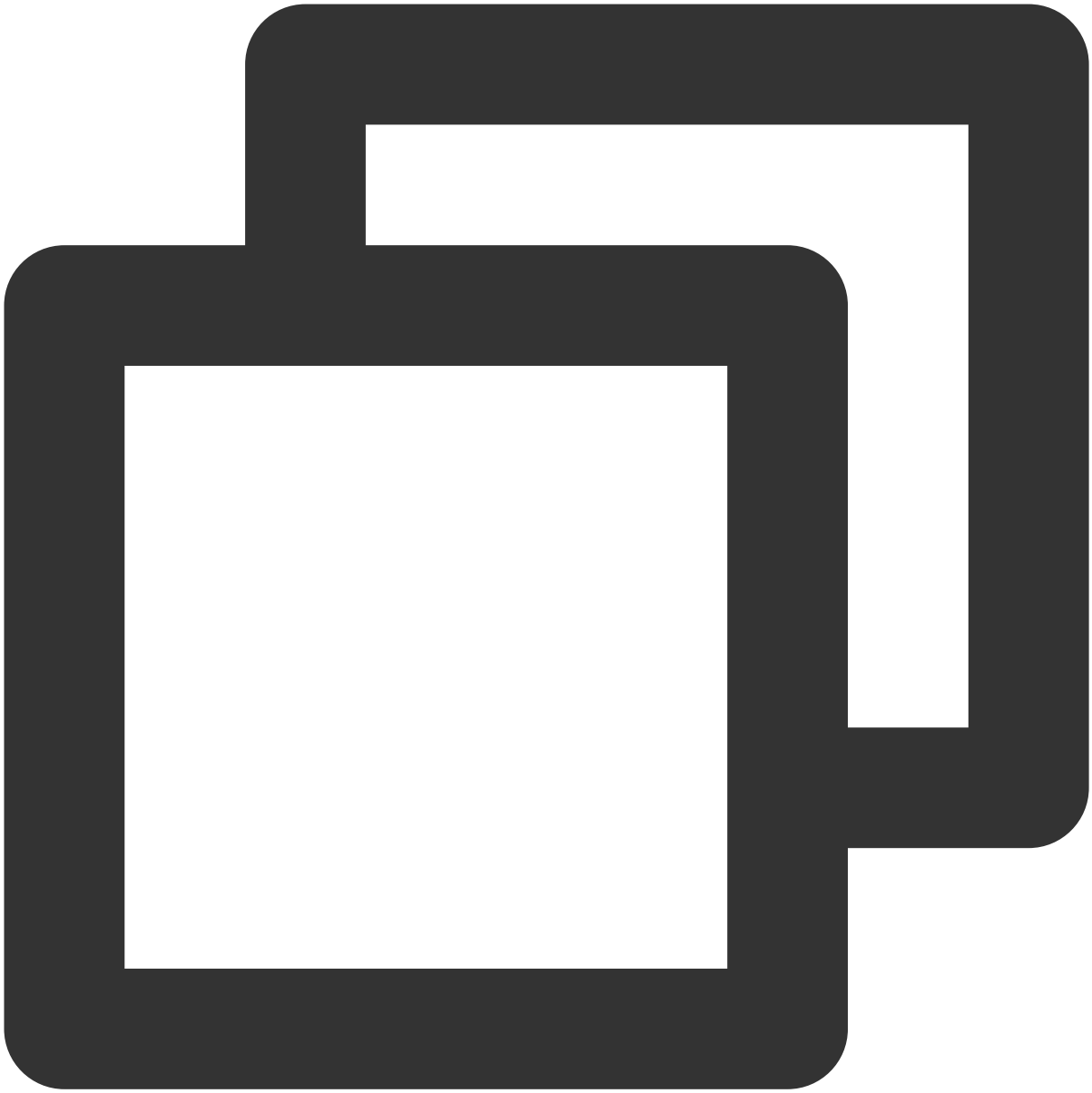
```
// 引入TCCC相关包
import {TcccWorkstation,TCCCLoginType,TCCCAudioRoute,TCCCEndReason} from "tccc-sdk-
// 创建实例和设置事件回调
const tcccSDK = TCCCWorkstation.sharedInstance();
// 错误事件回调
tcccSDK.on('onError',(errCode,errMsg) => {
});
// 通话结束回调
tcccSDK.on('onEnded',(reason,reasonMessage,sessionId) => {
    if (reason == 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.off('*');
```

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



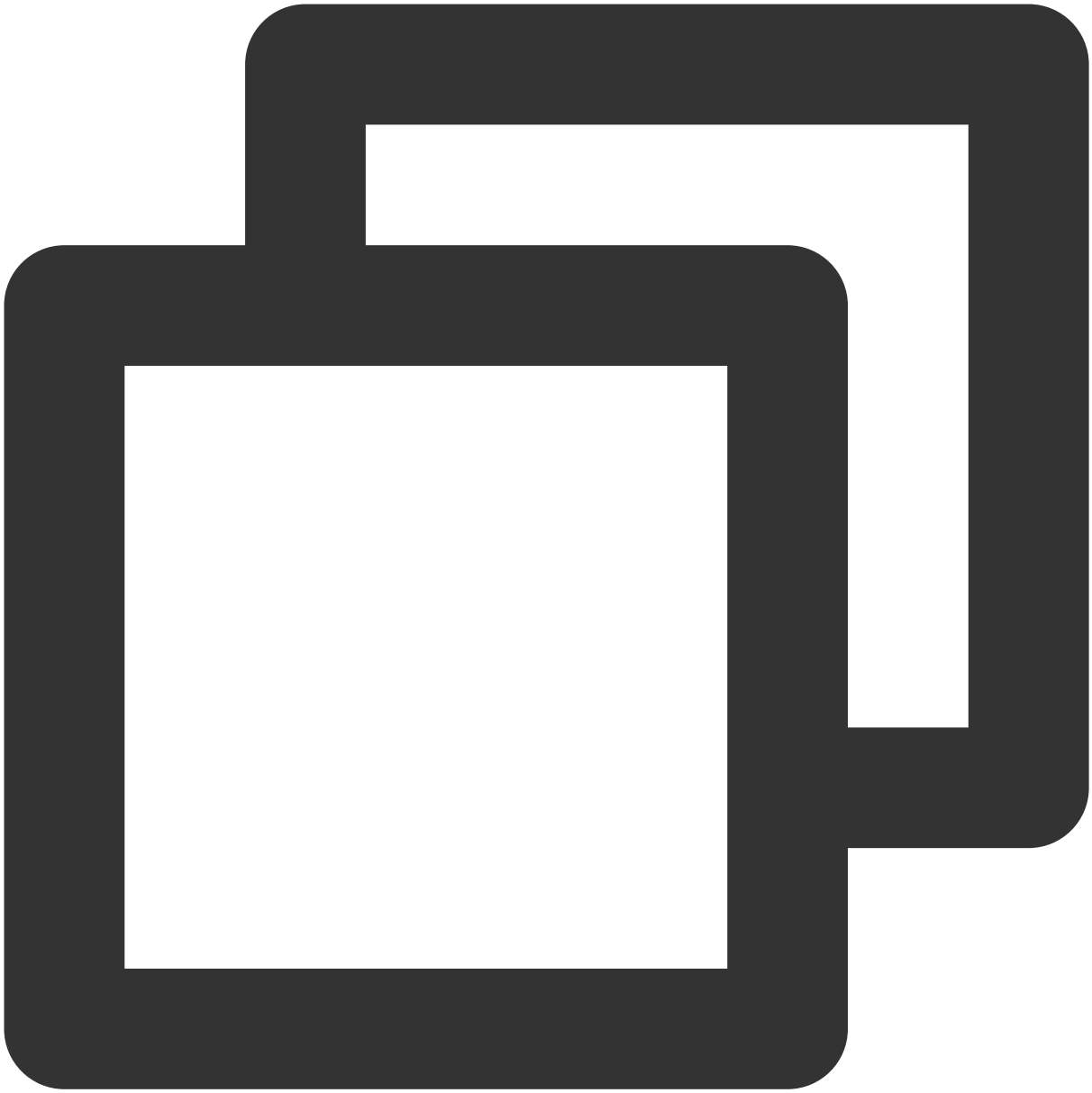
```
// 其中sdkAppId、userId、token的获取参考关键概念对应的字段。
// 座席登录
tcccSDK.login({
    sdkAppID: sdkAppId,
    userId: userID,
    token: token,
    type: type,
},(code,message) => {
    if (code == TcccErrorCode.ERR_NONE) {
        // 登录成功
    } else {
```

```
        // 登录失败
    }
});
// 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
tcccSDK.checkLogin(code,message) => {
    if (code == TcccErrorCode.ERR_NONE) {
        // 已登录
    } else {
        // 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

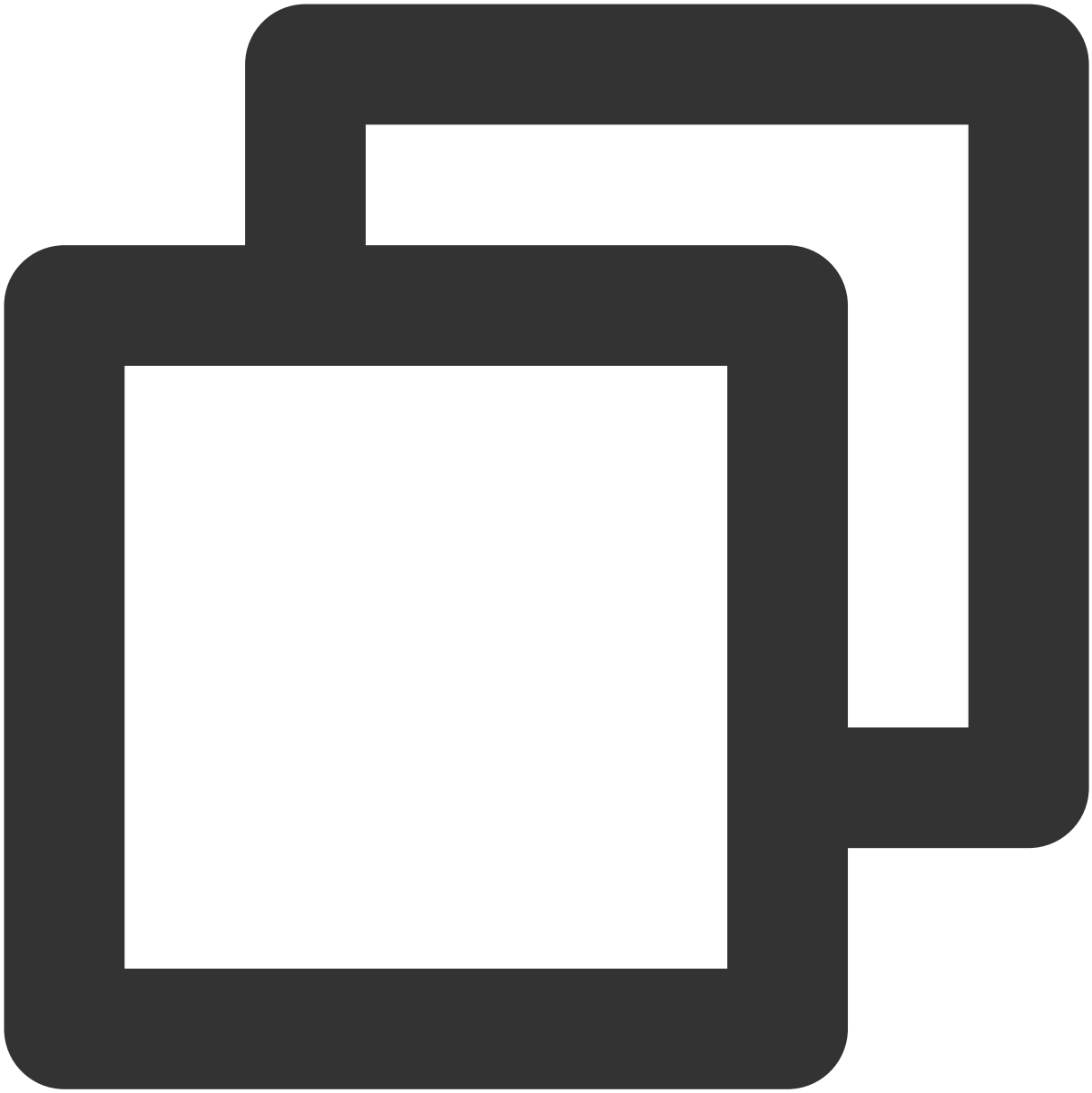


```
// 发起呼叫
tcccSDK.call({
  to: '134xxxx',          // 被叫号码（必填）
  remark: "xxx",          // 号码备注，在通话条中会替代号码显示（可选）
  uui: "xxxx",            // 户自定义数据（可选）
}, (code,message) => {
  if (code == TcccErrorCode.ERR_NONE) {
    // 发起成功
  } else {
    // 发起失败
  }
```

```
});

// 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
    }
});
```

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



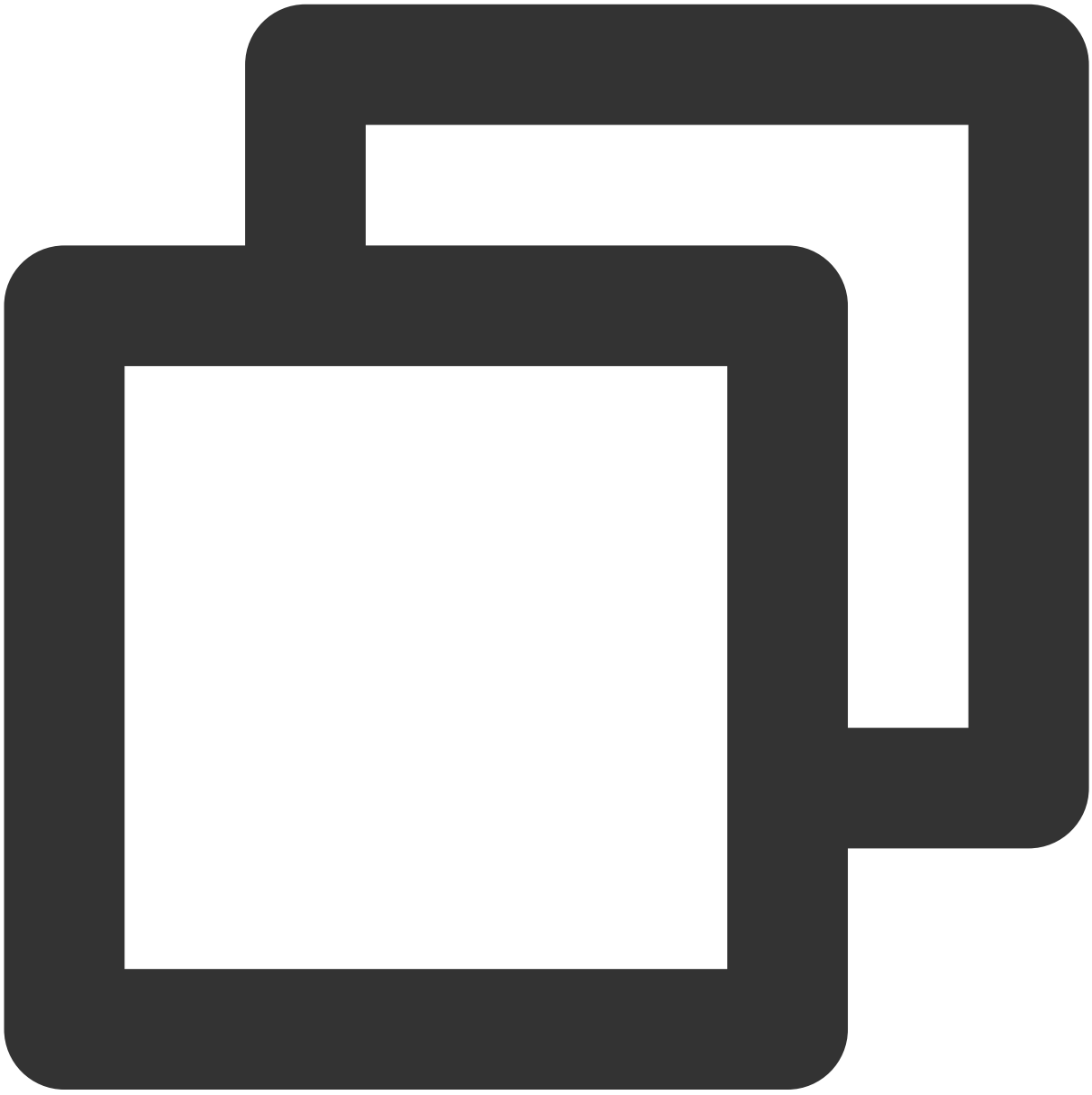
```
// TCCCAudioRoute.Earpiece 为耳麦
// 设置为扬声器
const route = TCCCAudioRoute.Speakerphone;
tcccSDK.getDeviceManager().setAudioRoute(route);
```

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

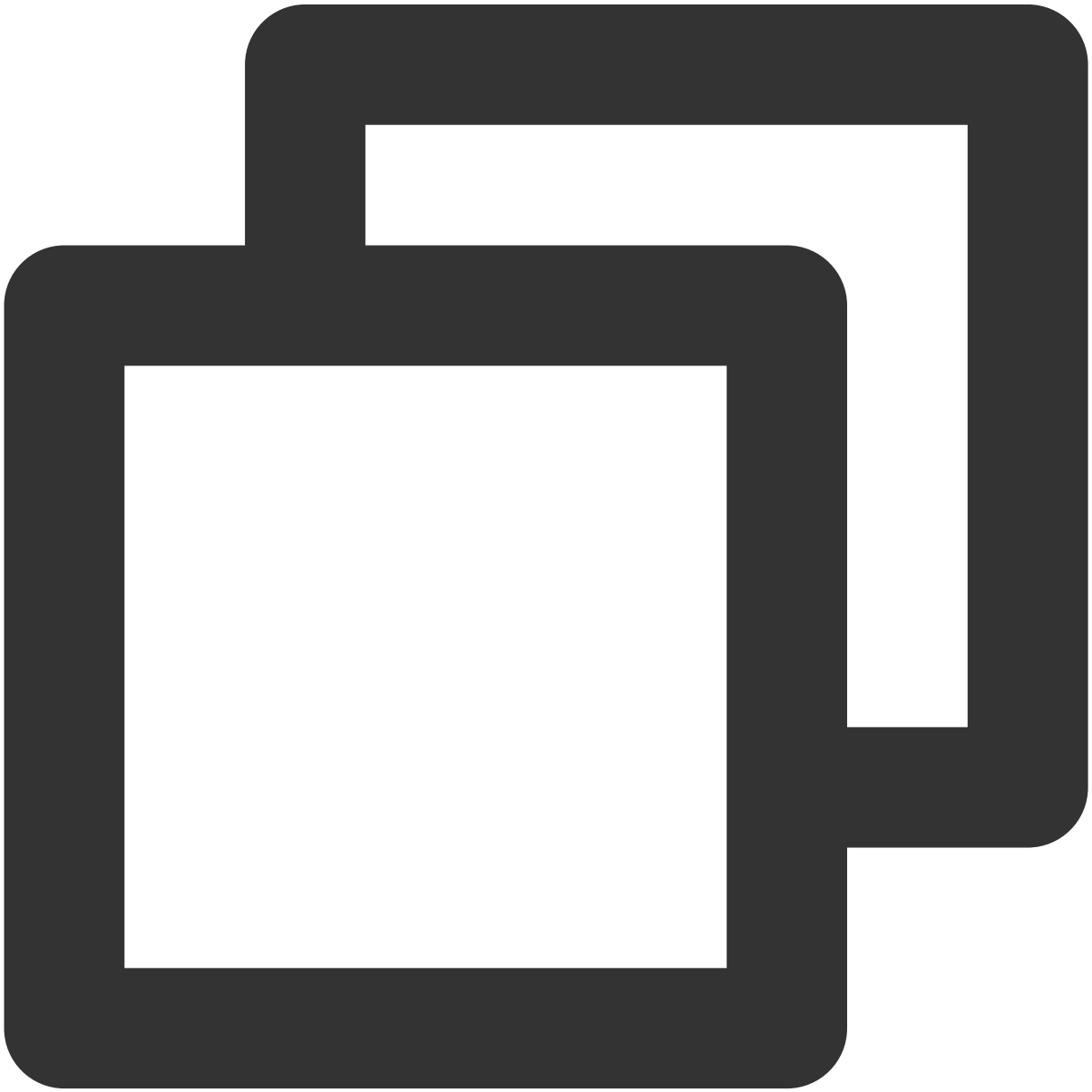


```
// 获取SDK 版本号
TCCCWorkstation.getSDKVersion();
```

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



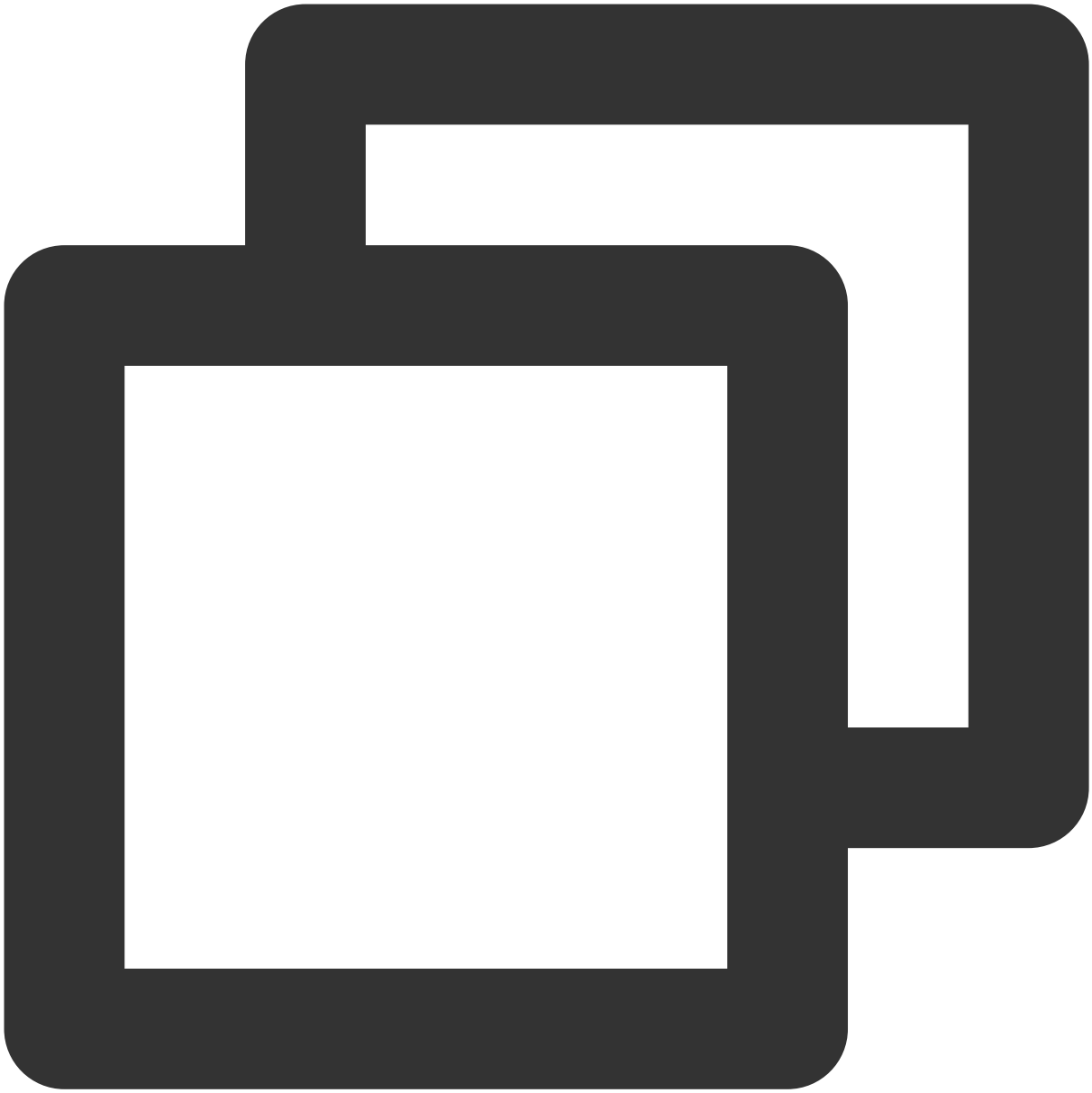
```
// 错误事件回调
```

```
tcccSDK.on('onError',(errCode,errMsg) => {
});
// 警告事件回调
tcccSDK.on('onWarning',(warningCode,warningMsg) => {
});
```

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



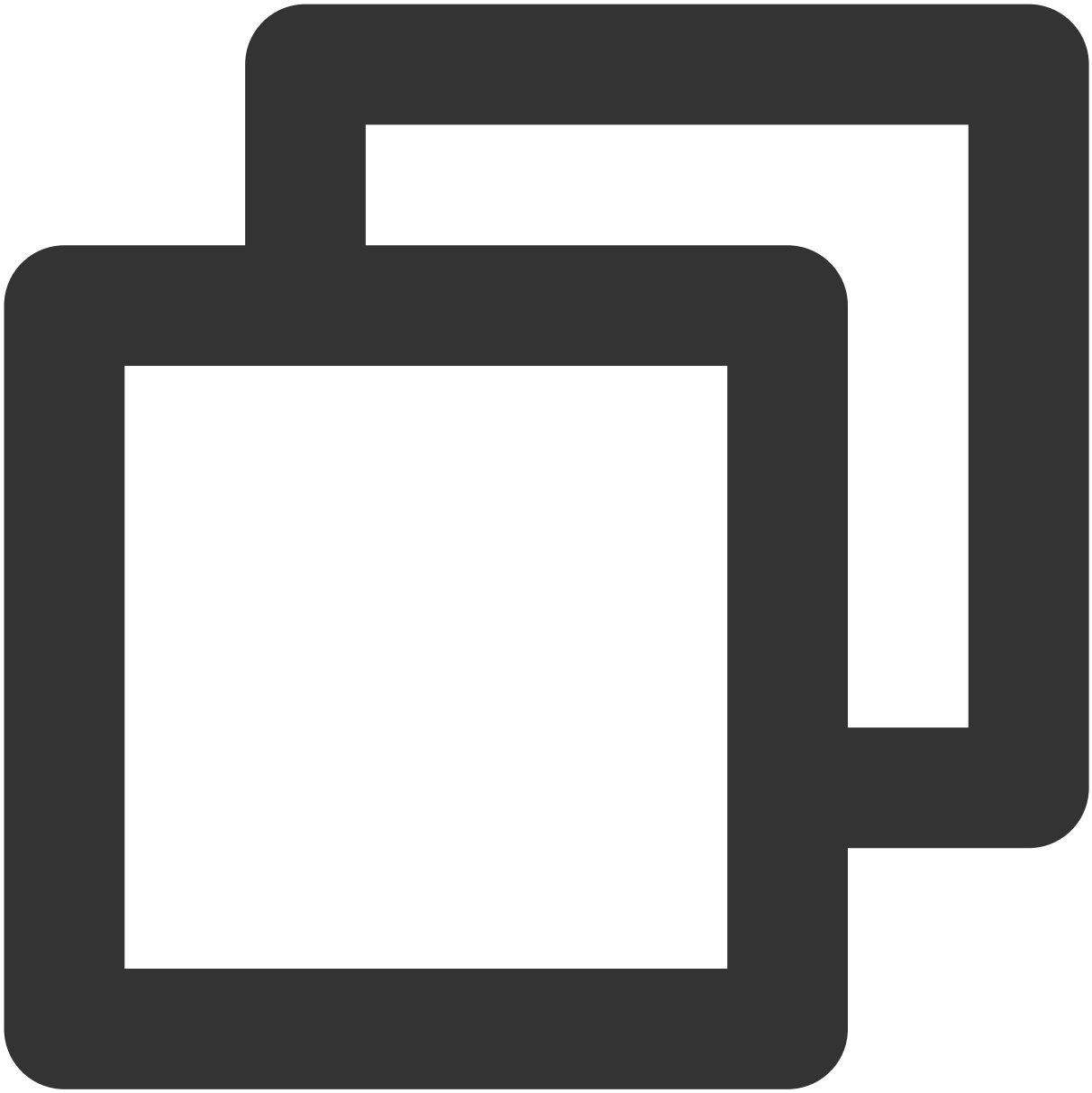
```
// 会话结束事件
tcccSDK.on("onEnded",(reason,reasonMessage,sessionId) => {
    var msg = reasonMessage;
    if (reason == TCCCEndReason.Error) {
        msg = "系统异常"+reasonMessage;
    } else if (reason == TCCCEndReason.Timeout) {
        msg = "超时挂断";
    } else if (reason == TCCCEndReason.LocalBye) {
        msg = "您已挂断";
    } else if (reason == TCCCEndReason.RemoteBye) {
        msg = "对方已挂断";
```

```
    } else if (reason == TCCCEndReason.Rejected) {
        msg = "对方已拒接";
    } else if (reason == TCCCEndReason.RemoteCancel) {
        msg = "对方已取消";
    }
});
// 新会话事件。包括呼入和呼出
tcccSDK.on('onNewSession',(res) => {
    const sessionDirection = res.sessionDirection;
    if (sessionDirection == TCCCSessionDirection.CallIn) {
        // 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
    } else if (sessionDirection == TCCCSessionDirection.CallOut){
        // 呼出
    }
});
// 对端已接听
tcccSDK.on('onAccepted',(sessionId) => {
});
// 网络质量的实时统计回调
tcccSDK.on('onNetworkQuality',(localQuality) => {
    const quality = localQuality.quality;
//   ///当前网络一般
//   TCCCQuality_Poor = 3,
//   ///当前网络较差
//   TCCCQuality_Bad = 4,
//   ///当前网络很差
//   TCCCQuality_Vbad = 5,
//   ///当前网络不满足 通话 的最低要求
//   TCCCQuality_Down = 6,

});
// 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_SAMPLERATE	-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_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_REFUSED	-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

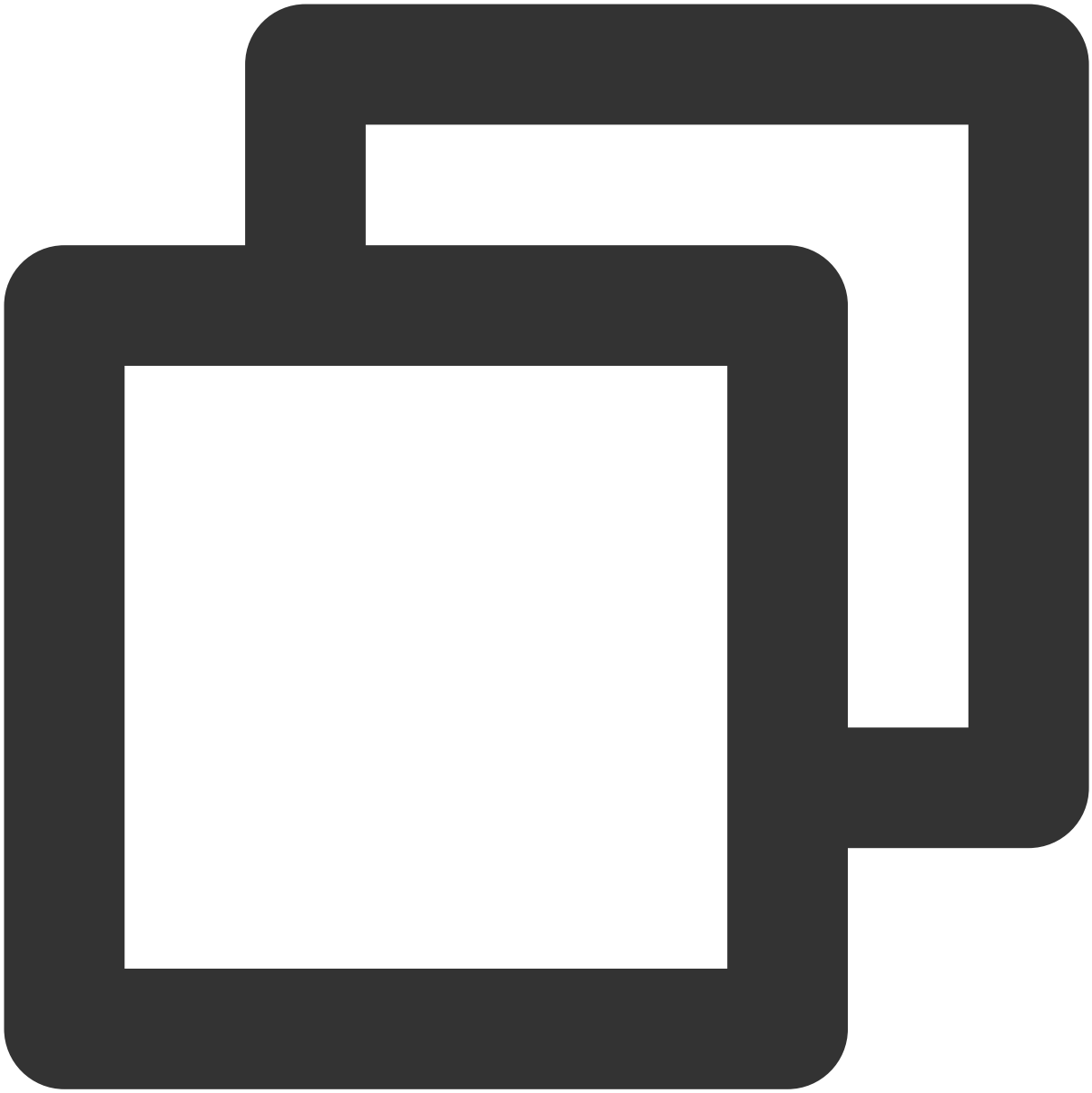
Android

最近更新时间：2024-04-01 17:50:58

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



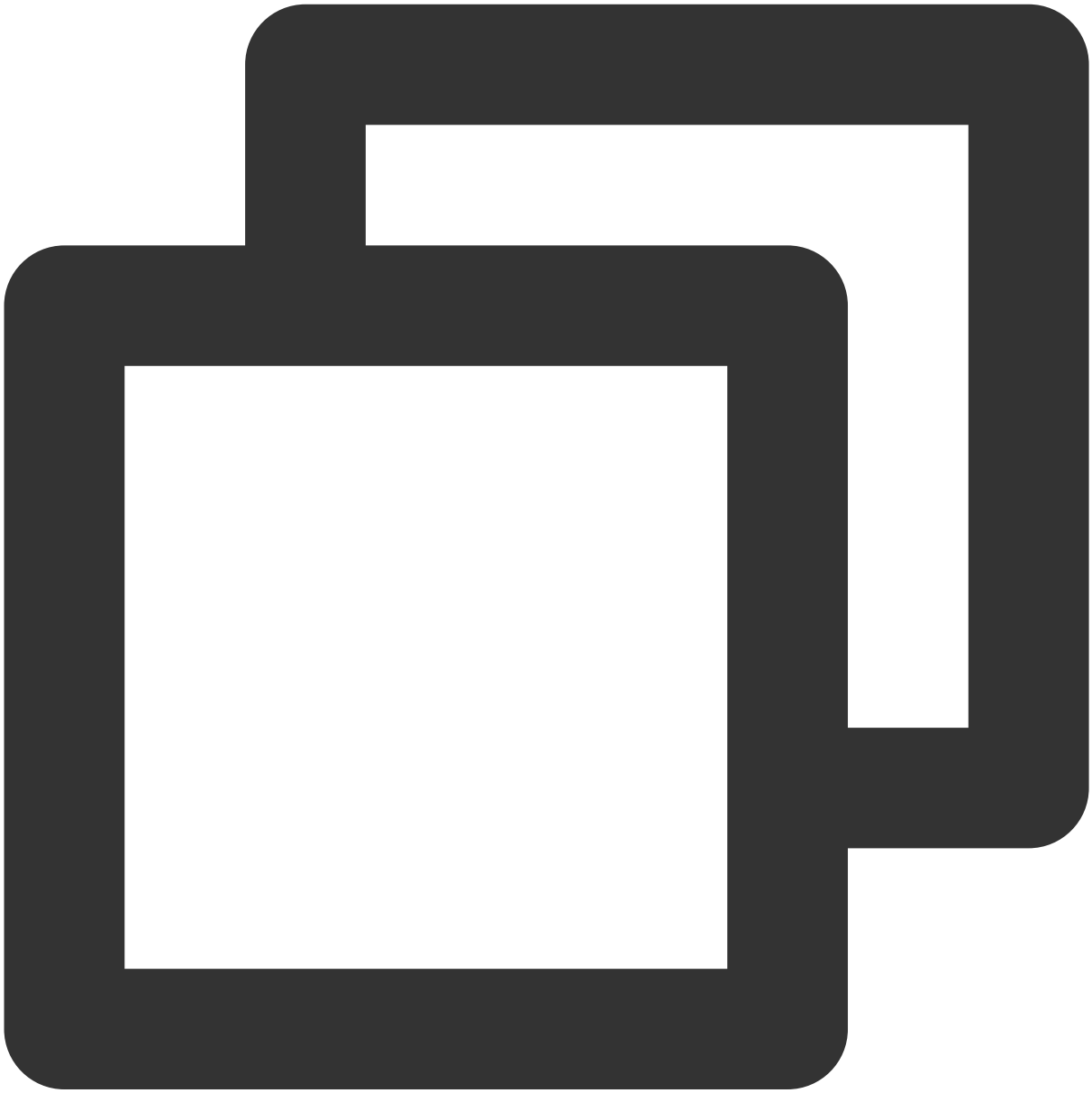
```
// 创建实例和设置事件回调
TCCCWorkstation tcccSDK = TCCCWorkstation.sharedInstance(getApplicationContext());
tcccSDK.setListener(new TCCCListener() {});
```

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



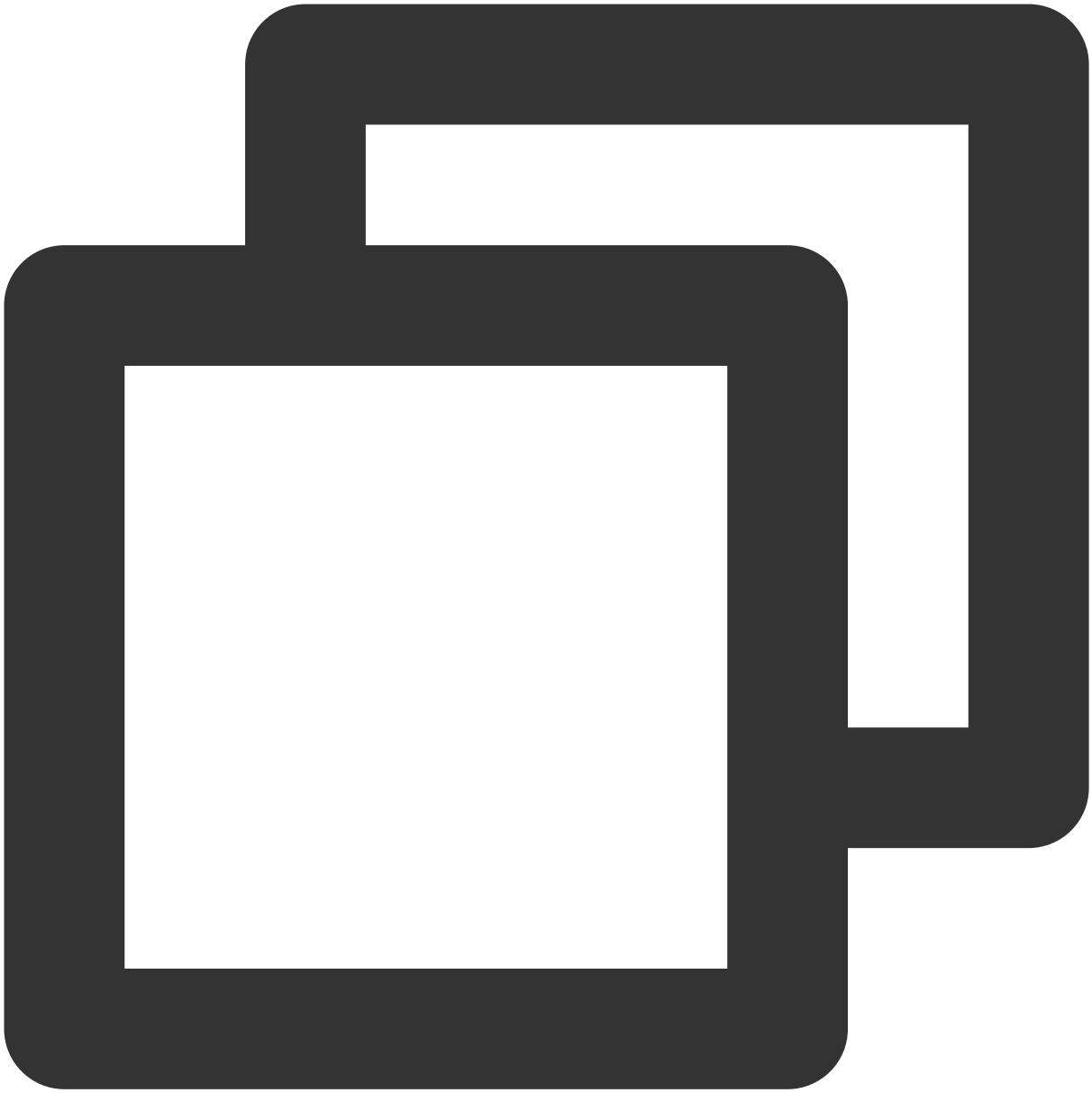
```
TCCCTypeDef.TCCCLoginParams loginParams = new TCCCTypeDef.TCCCLoginParams();
/// 登录的坐席ID，通常为邮箱地址
loginParams.userId = "";
/// 登录票据，在登录模式为Agent必填。更多详情请参见[创建 SDK 登录
/// Token](https://cloud.tencent.com/document/product/679/49227)
```

```
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) {
      // login error
    }
});
```

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();
//格式 <scheme> : <user> @<host>, 如 sip:1343xxxx@1400xxxx.tccc.qcloud.com, 其中1343x
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
    }
});
// 结束通话
tcccSDK.terminate();
```

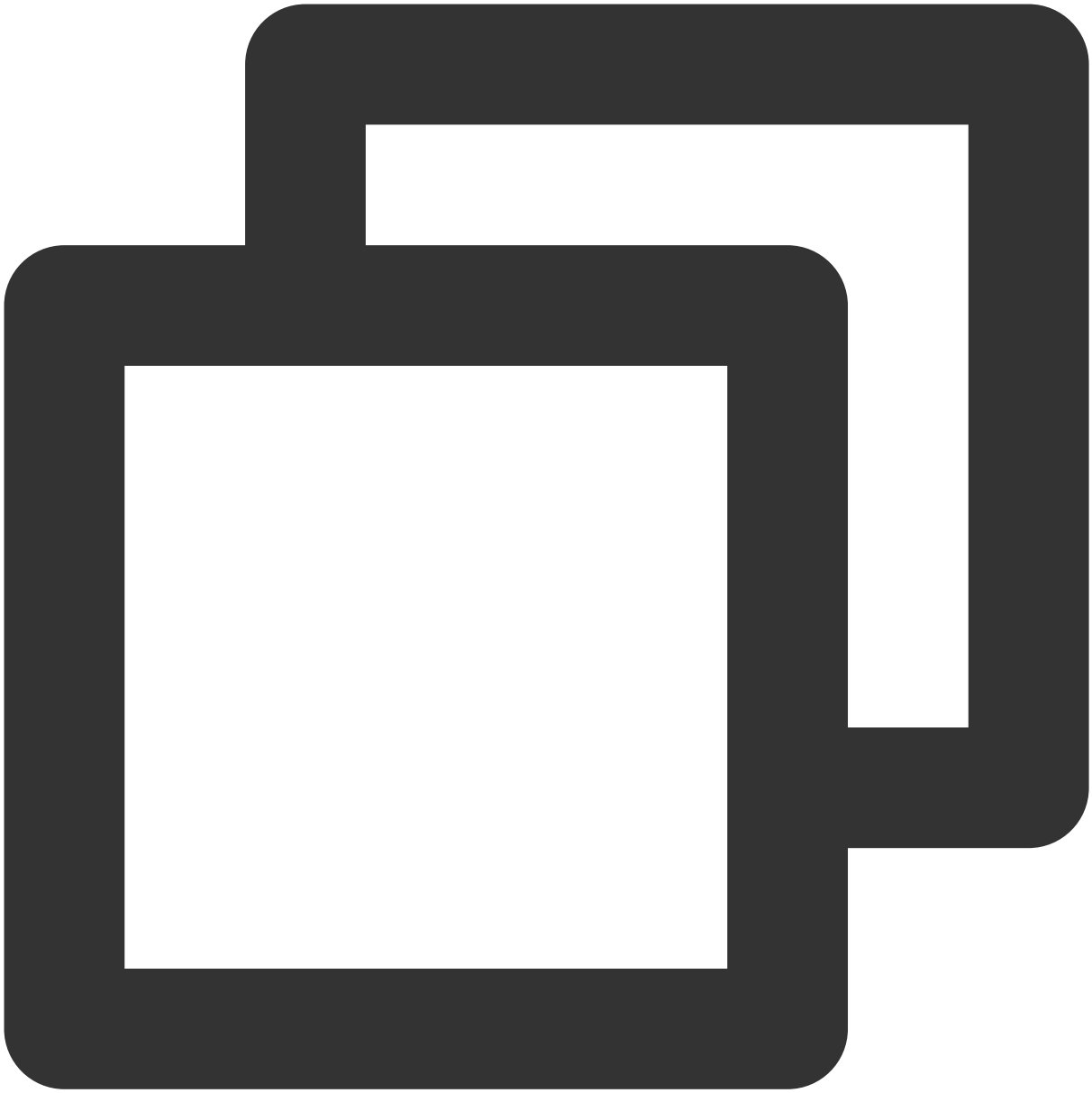
音频设备接口函数

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

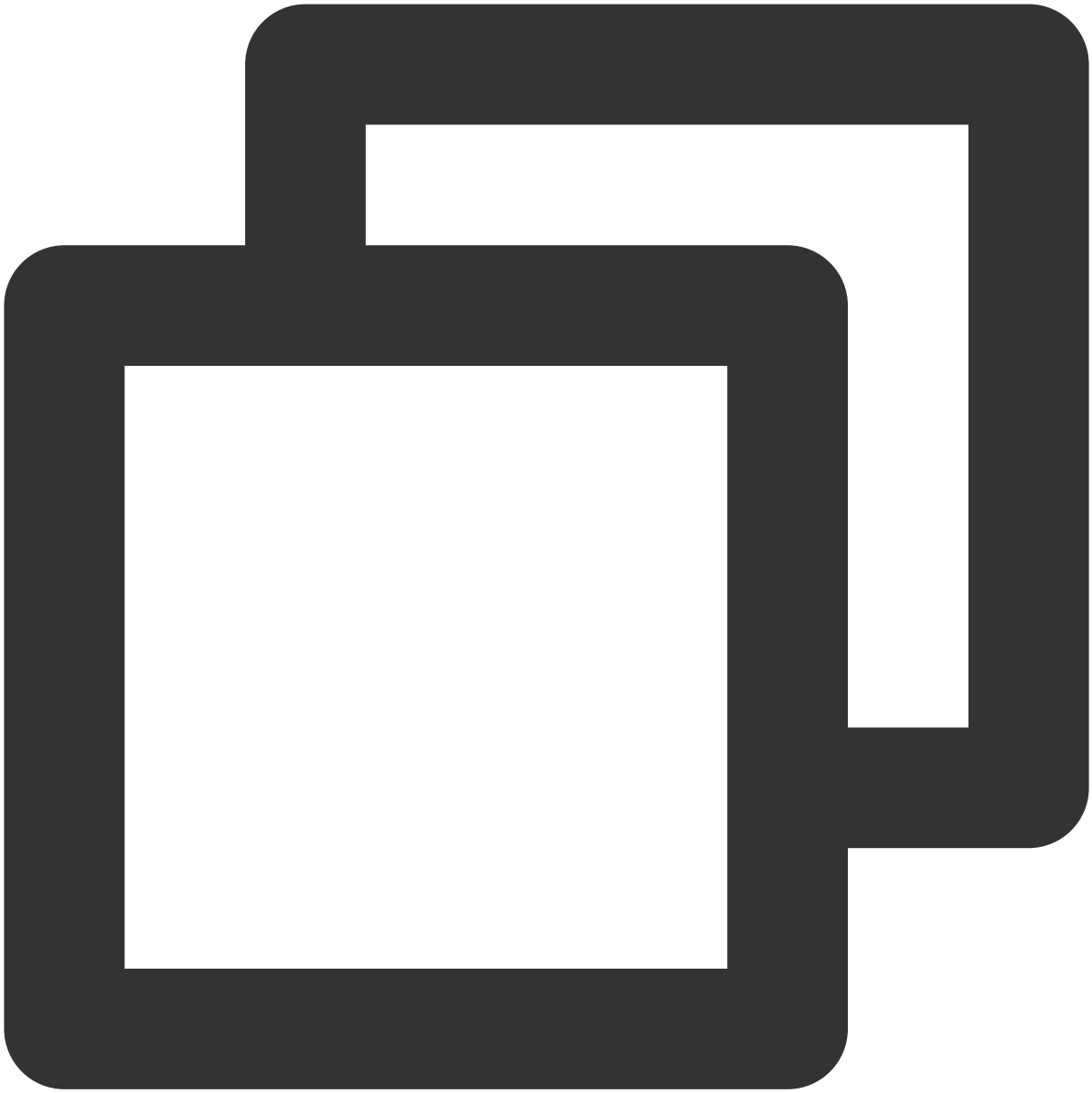


```
// 获取SDK 版本号
TCCCWorkstation.getSDKVersion();
```

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



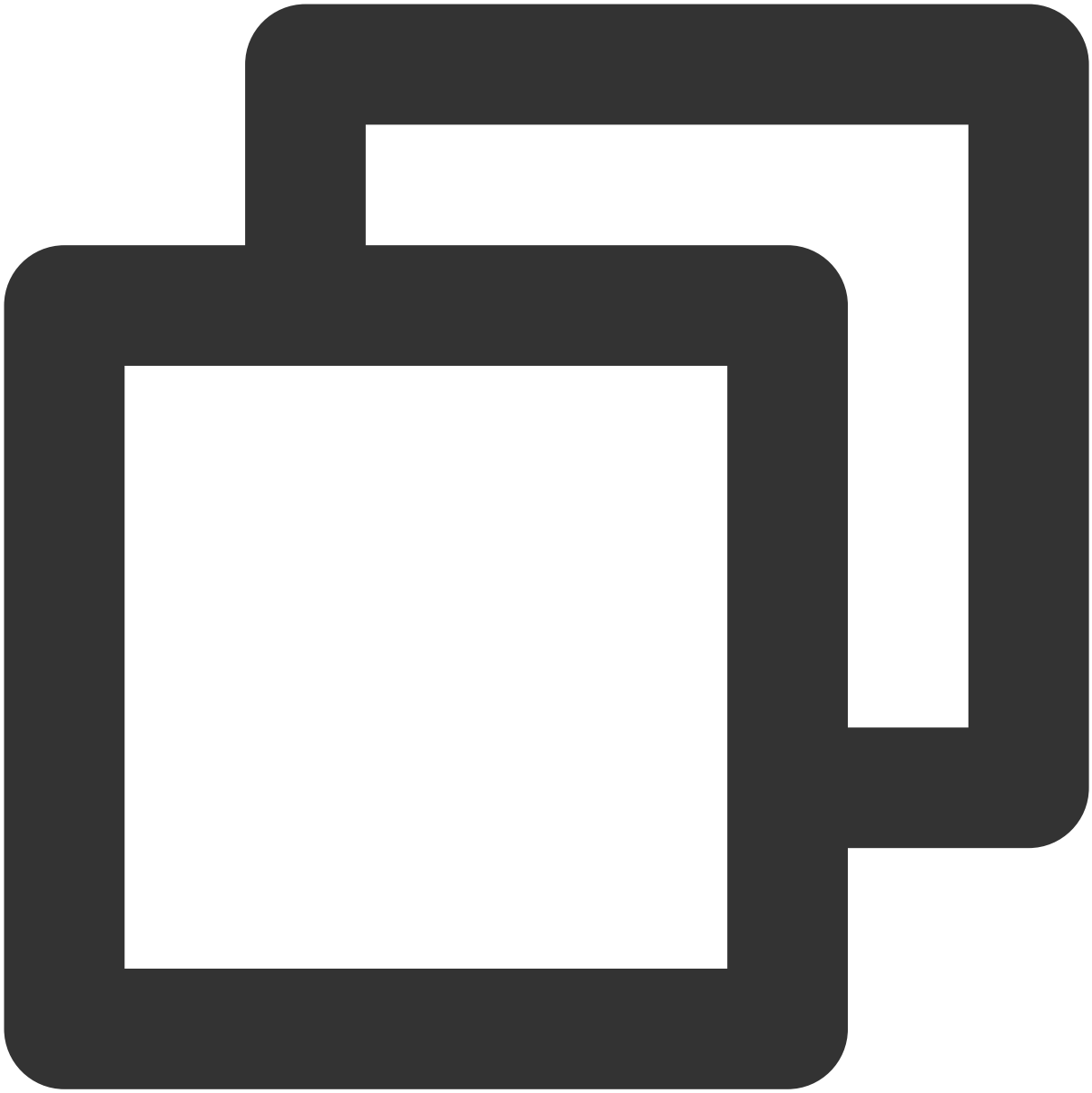
```
tcccSDK.setListener(new TCCCListener() {
    /**
         * 错误事件回调
         * 错误事件，表示 SDK 抛出的不可恢复的错误，比如进入房间失败或设备开启失败等。
         * @param errCode    错误码
         * @param errMsg     错误信息
         * @param extraInfo 扩展信息字段，个别错误码可能会带额外的信息帮助定位问题
         */
    @Override
```

```
    public void onError(int errCode, String errMsg, Bundle extraInfo) {
        super.onError(errCode, errMsg, extraInfo);
    }

    /**
        * 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 警告码
        * @param warningMsg  警告信息
        * @param extraInfo   扩展信息字段，个别警告码可能会带额外的信息帮助定位问题
        */
    @Override
    public void onWarning(int warningCode, String warningMsg, Bundle extraInfo) {
        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



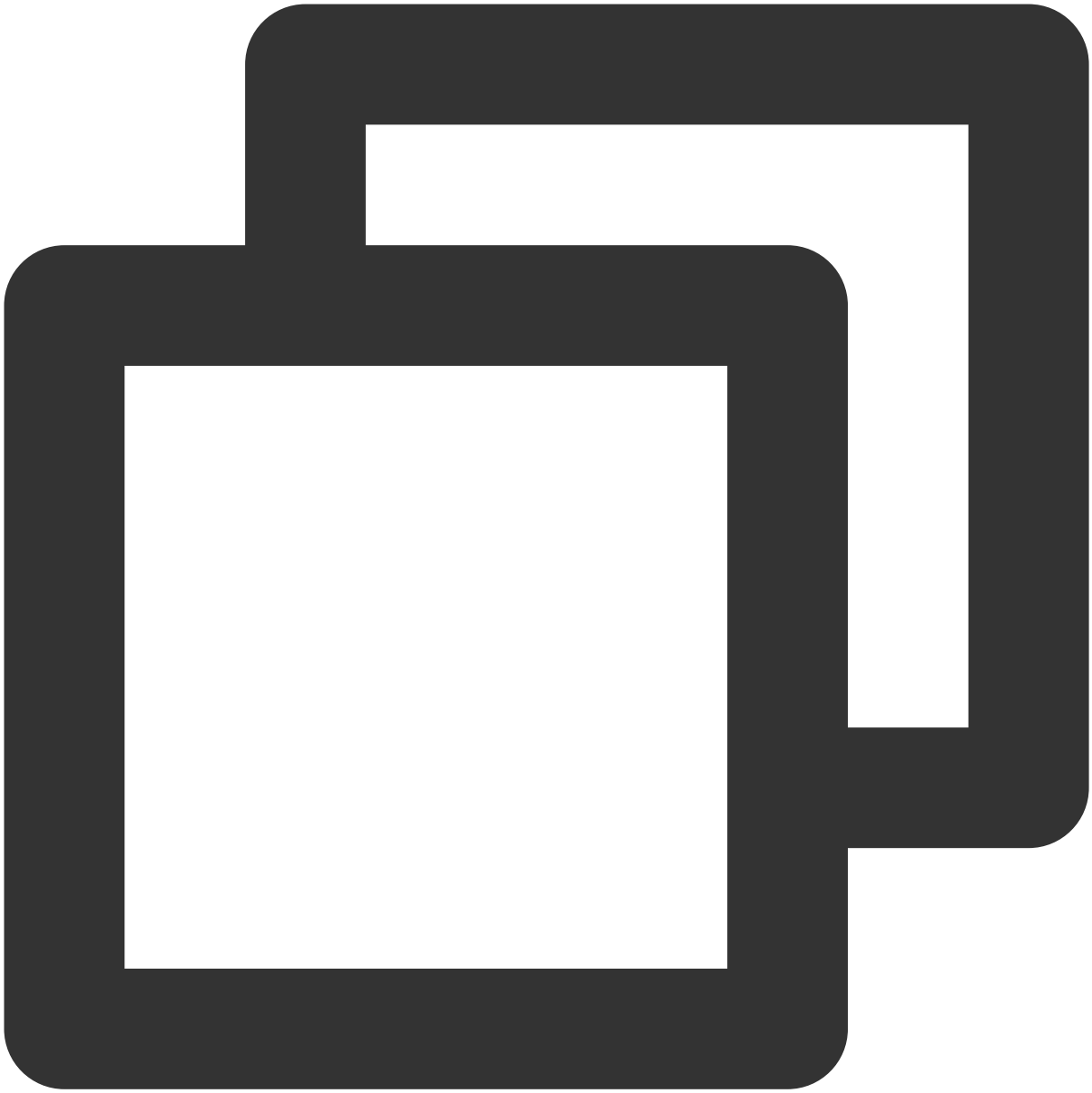
```
tcccSDK.setListener(new TCCCListener() {
    @Override
    public void onNewSession(TCCCTypeDef.ITCCCSessionInfo info) {
      super.onNewSession(info);
      // 新会话事件。包括呼入和呼出，可通过 info.sessionDirection 判断是呼入还是呼出
    }

    @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 与云端的连接已经断开
         * 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
         * 重连过程中会抛出 onTryToReconnect，连接恢复后会抛出 onConnectionRecovery 。
         * 所以，SDK 会在如下三个连接相关的事件中按如下规律切换：
         */
    @Override
    public void onConnectionLost(TCCCServerType serverType) {
        super.onConnectionLost(serverType);
```

```
    }

    /**
         * SDK 正在尝试重新连接到云端
         * 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 onConnectionRecovery。
         */
    @Override
    public void onTryToReconnect(TCCCServerType serverType) {
        super.onTryToReconnect(serverType);
    }

    /**
         * SDK 与云端的连接已经恢复
         * 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。
         */
    @Override
    public void onConnectionRecovery(TCCCServerType serverType) {
        super.onConnectionRecovery(serverType);
    }
});
```

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

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_AUTHENTICATION_REQUIRED	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_SAMPLERATE	-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_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_REFUSED	-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：2024-04-01 17:52:01

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、OC、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

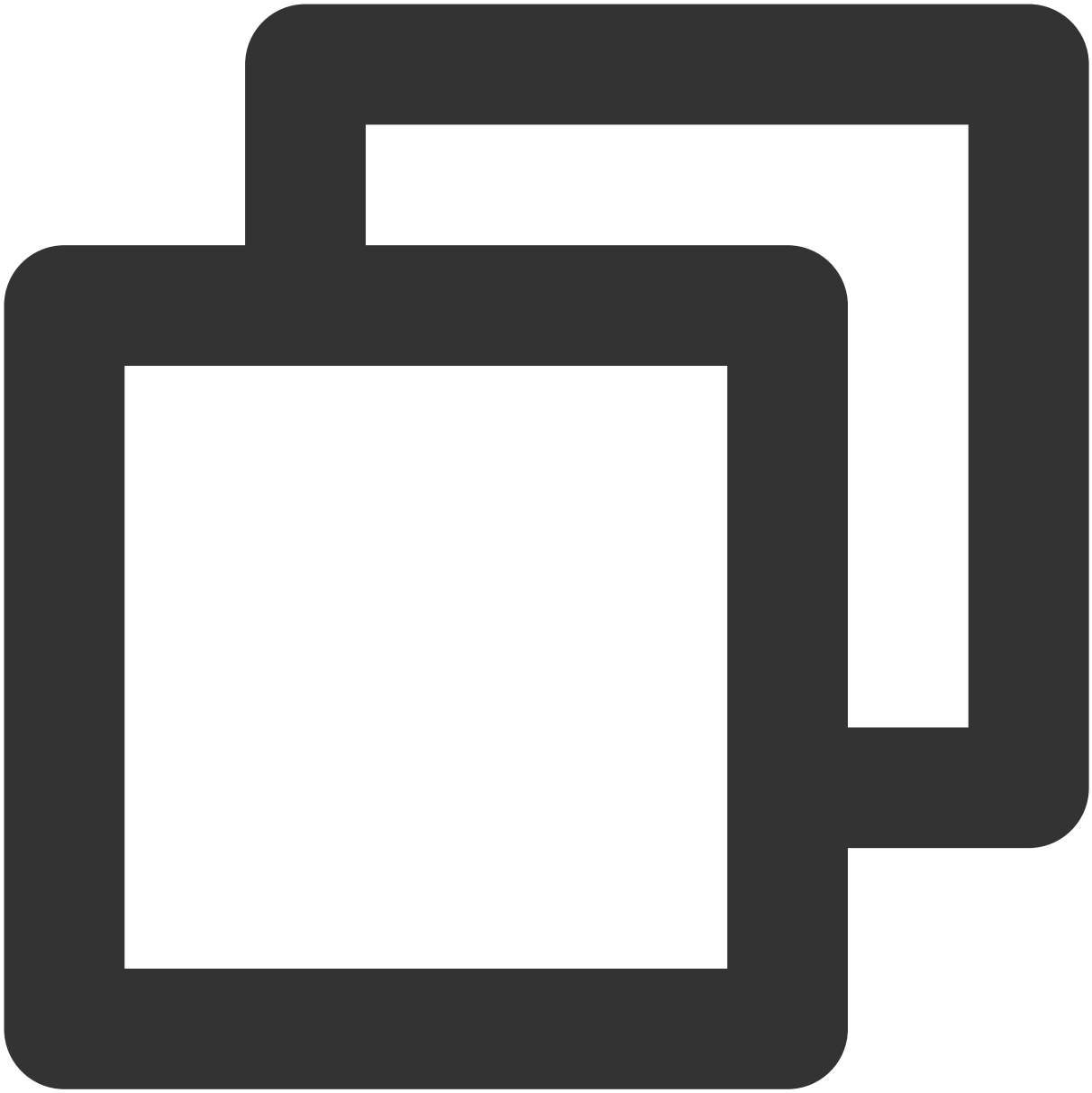
API	描述
sharedInstance	创建 ITCCCWorkstation 实例（单例模式）。
destroySharedInstance	销毁 ITCCCWorkstation 实例（单例模式）。
addTcccListener	添加 ITCCCWorkstation 事件回调。
removeTCCCListener	移除 ITCCCWorkstation 事件回调。

创建实例和设置事件回调示例代码

Swift

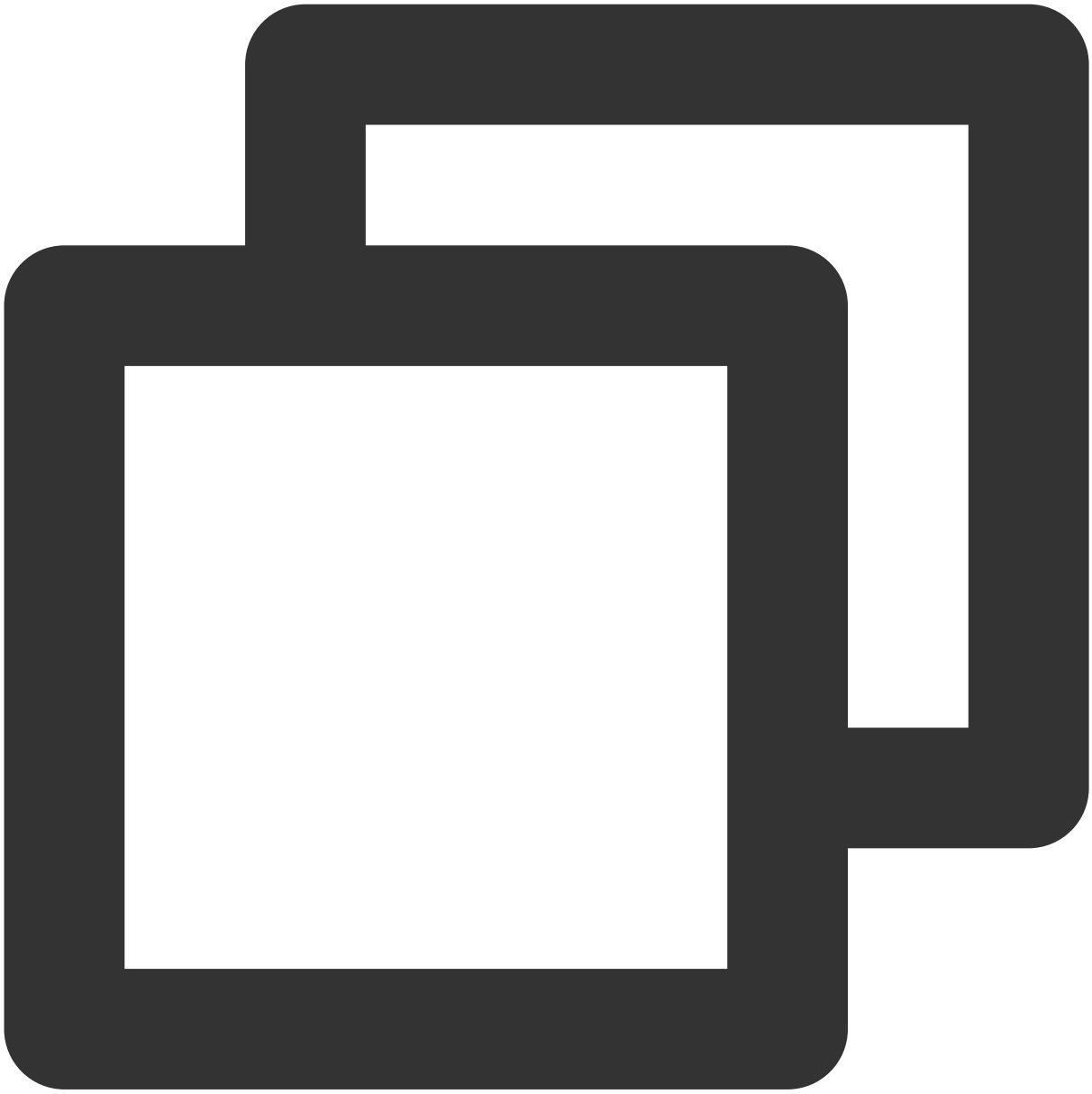
Objective-C

C++



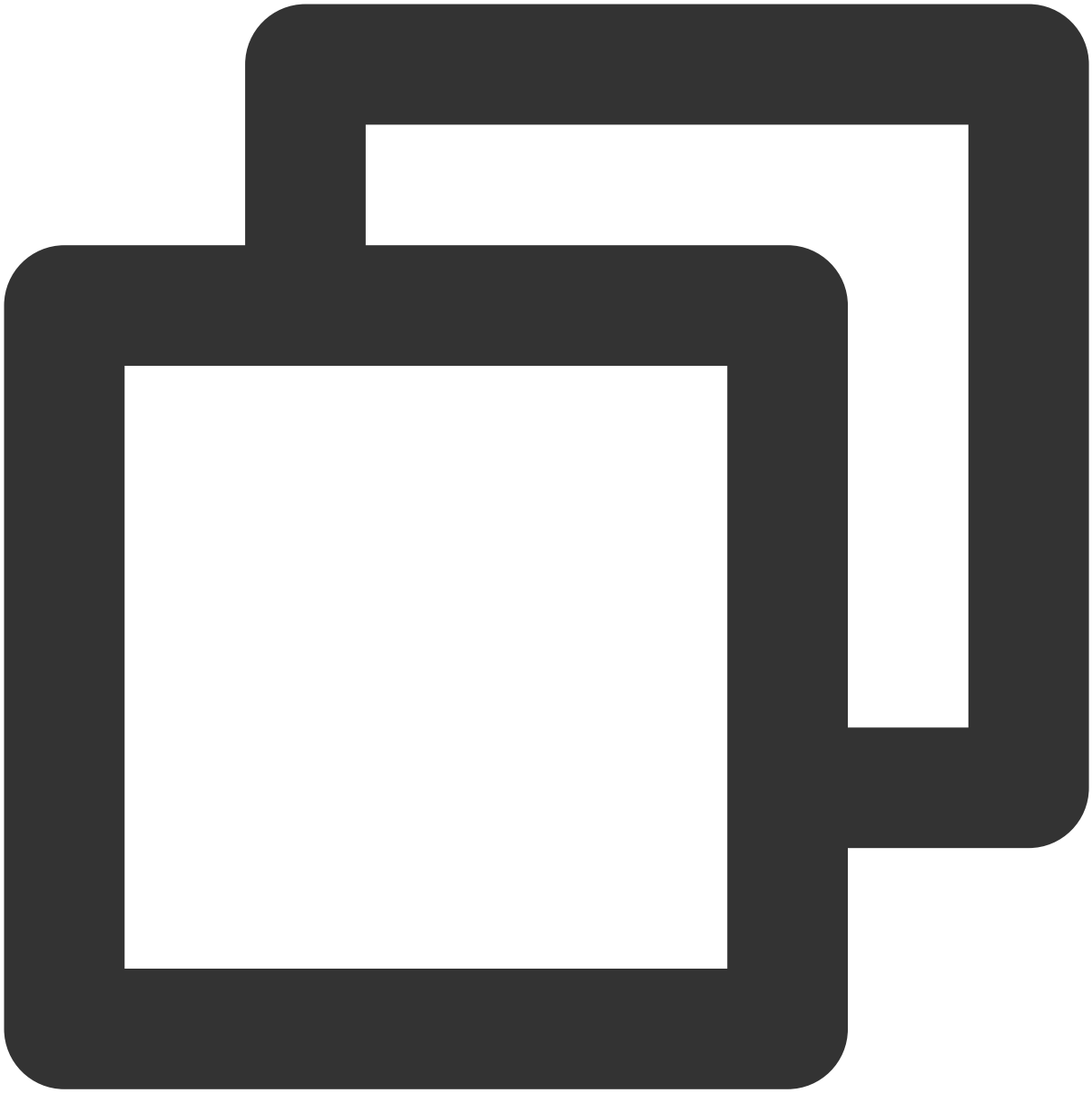
```
import TCCCSDK

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)
// 销毁实例
TCCCWorkstation.destroySharedIntance()
```



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

@property (strong, nonatomic) TCCCWorkstation *tcccSDK;

- (TCCCWorkstation*)tcccSDK {
    if (!_tcccSDK) {
```

```
        // 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;
```



```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
// 创建实例和设置事件回调
ITCCCWorkstation* tcccSDK = getTCCCShareInstance();
// 设置回调，TCCCCallbackImpl 需要继承 ITCCCCallback
class TCCCCallbackImpl:public ITCCCCallback {
public:
    TCCCCallbackImpl() {}
    ~TCCCCallbackImpl() {}

    void onError(TCCCError errCode, const char* errMsg, void* extraInfo) {}
```

```
    void onWarning(TCCCCWarning warningCode, const char* warningMsg, void* extraInf

    void onNewSession(TCCCSessionInfo info) {}

    void onEnded(EndedReason reason, const char* reasonMessage, const char* session

};
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();
tcccSDK->addCallback(tcccCallback);
// 销毁实例
destroyTCCCShareInstance();
tcccSDK = nullptr;
```

登录相关接口

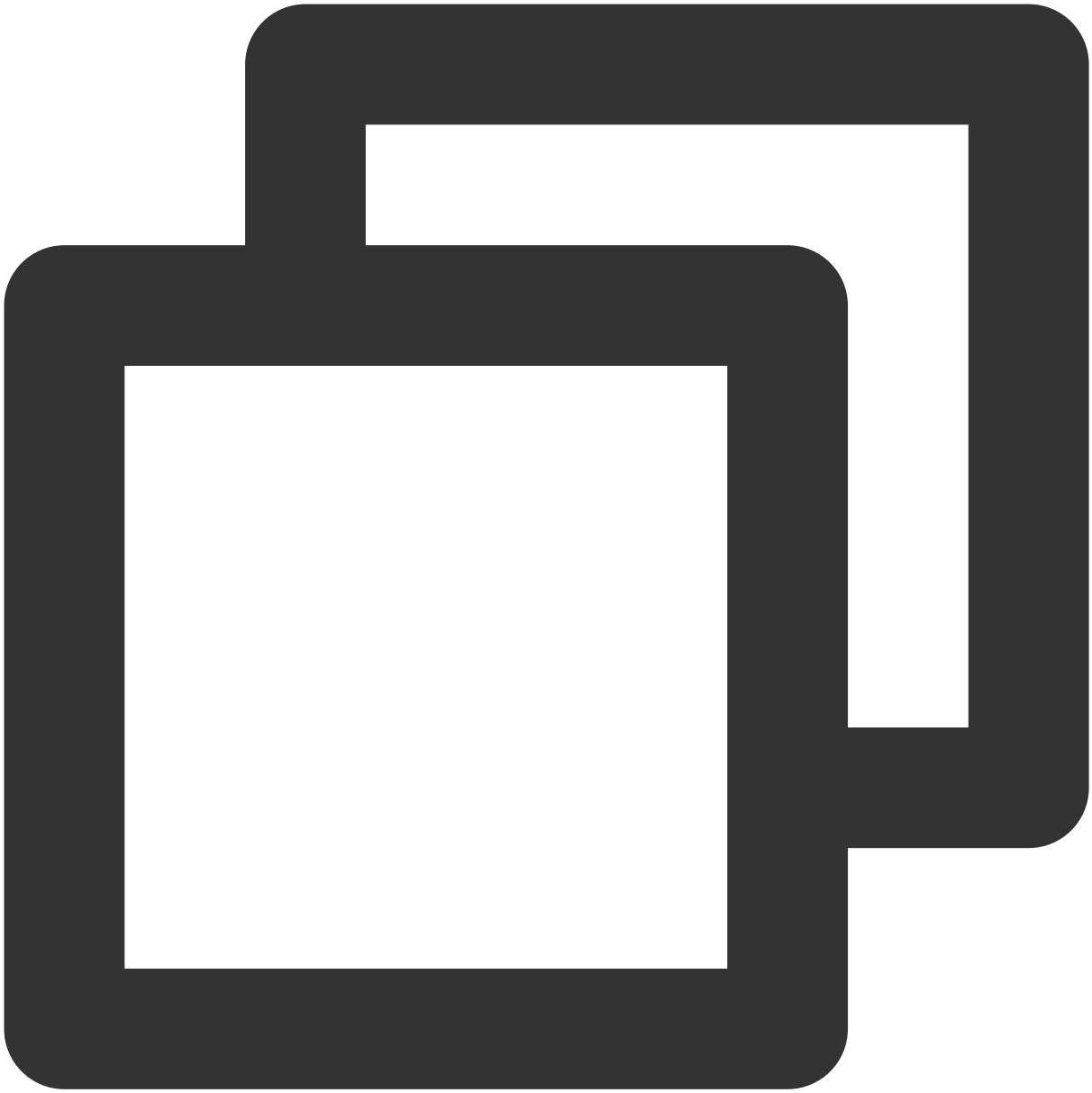
API	描述
login	SDK 登录。
checkLogin	检查 SDK 是否已登录。
logout	SDK 退出登录。

登录、退出登录示例代码

Swift

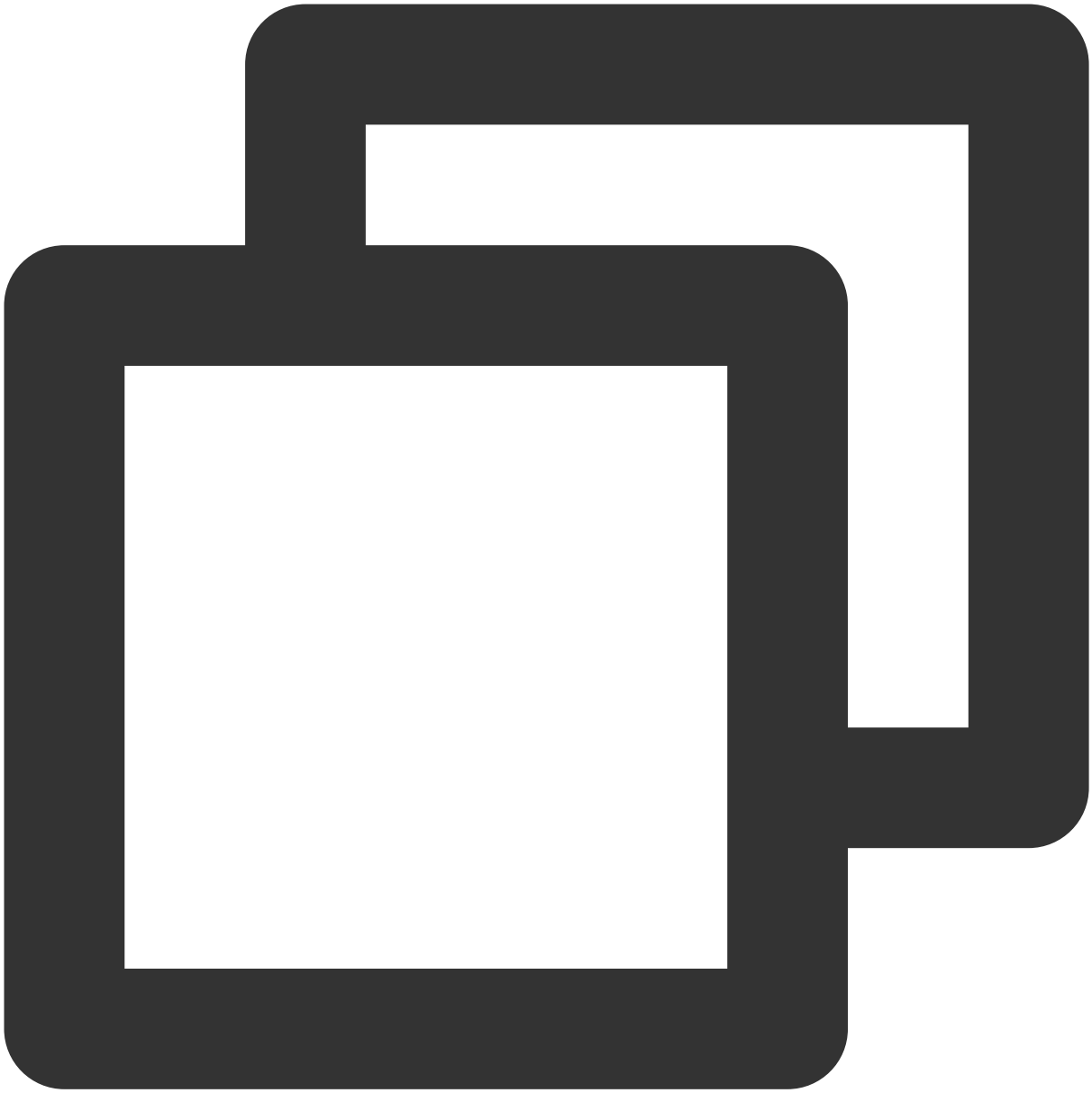
Objective-C

C++



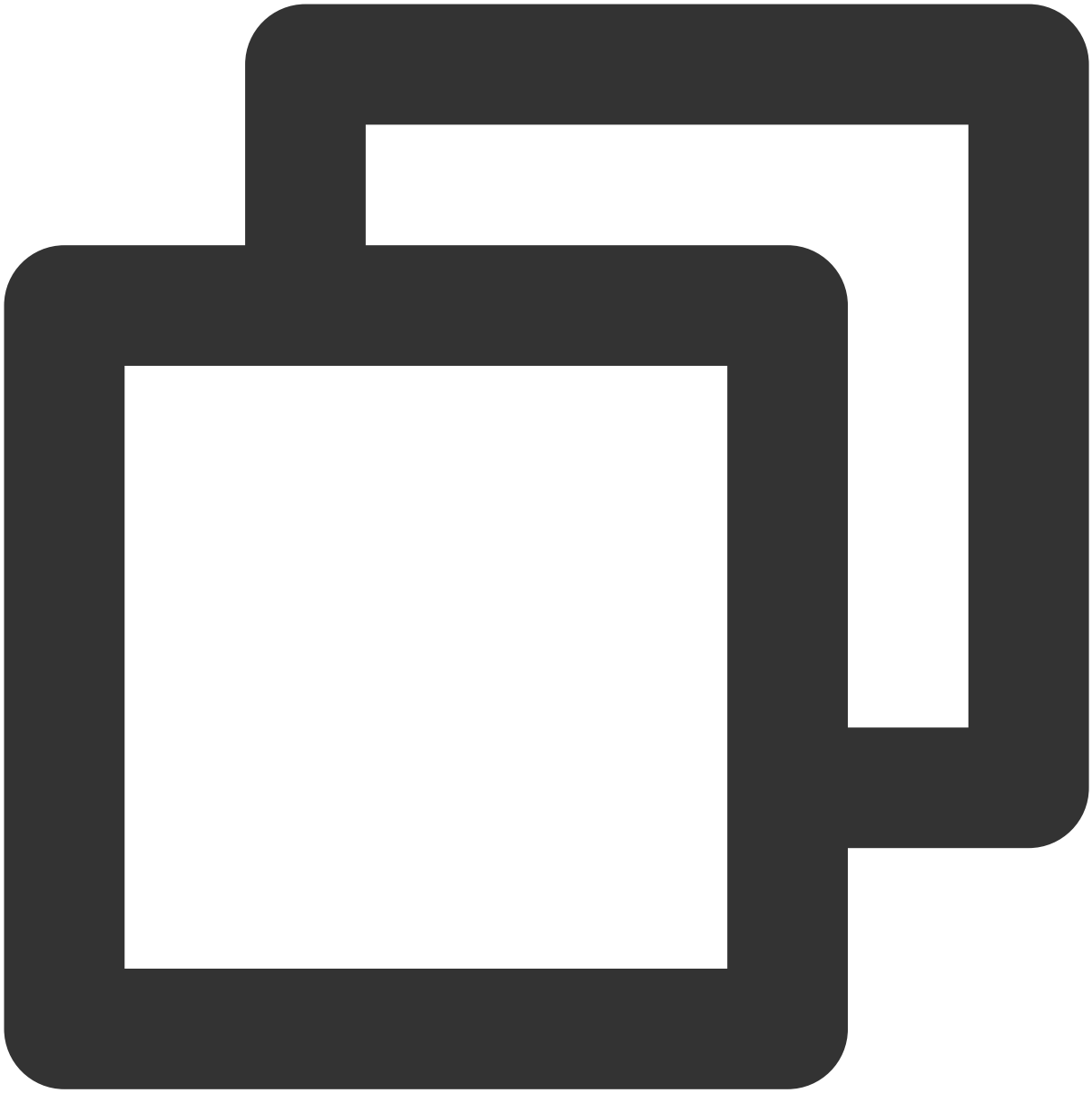
```
import TCCCSDK

let param = TXLoginParams()
// 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;
// 登录
tcccSDK.login(param) { info in
    // 登录成功
} fail: { code, message in
    // 登录失败
}

// 检查登录状态
tcccSDK.checkLogin {
  // 已登录
} fail: { code, message in
  // 未登录或者被T了
}

// 退出登录
tcccSDK.logout {
  // 退出成功
} fail: { code, message in
  // 退出异常
}
```



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

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T了
}];

// 退出登录
[self.tcccSDK logout:^{
    // 退出成功
} fail:^(int code, NSString * _Nonnull desc) {
    // 退出异常
}];
```



```
#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) {
  loginCallBackImpl = new TCCCLoginCallbackImpl();
}
TCCCLoginParams param;
/// 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 = TCCCLoginType::Agent;
// 登录
tcccSDK->login(param,loginCallBackImpl);
// 退出登录
tcccSDK->logout(nullptr);
```

呼叫相关接口函数

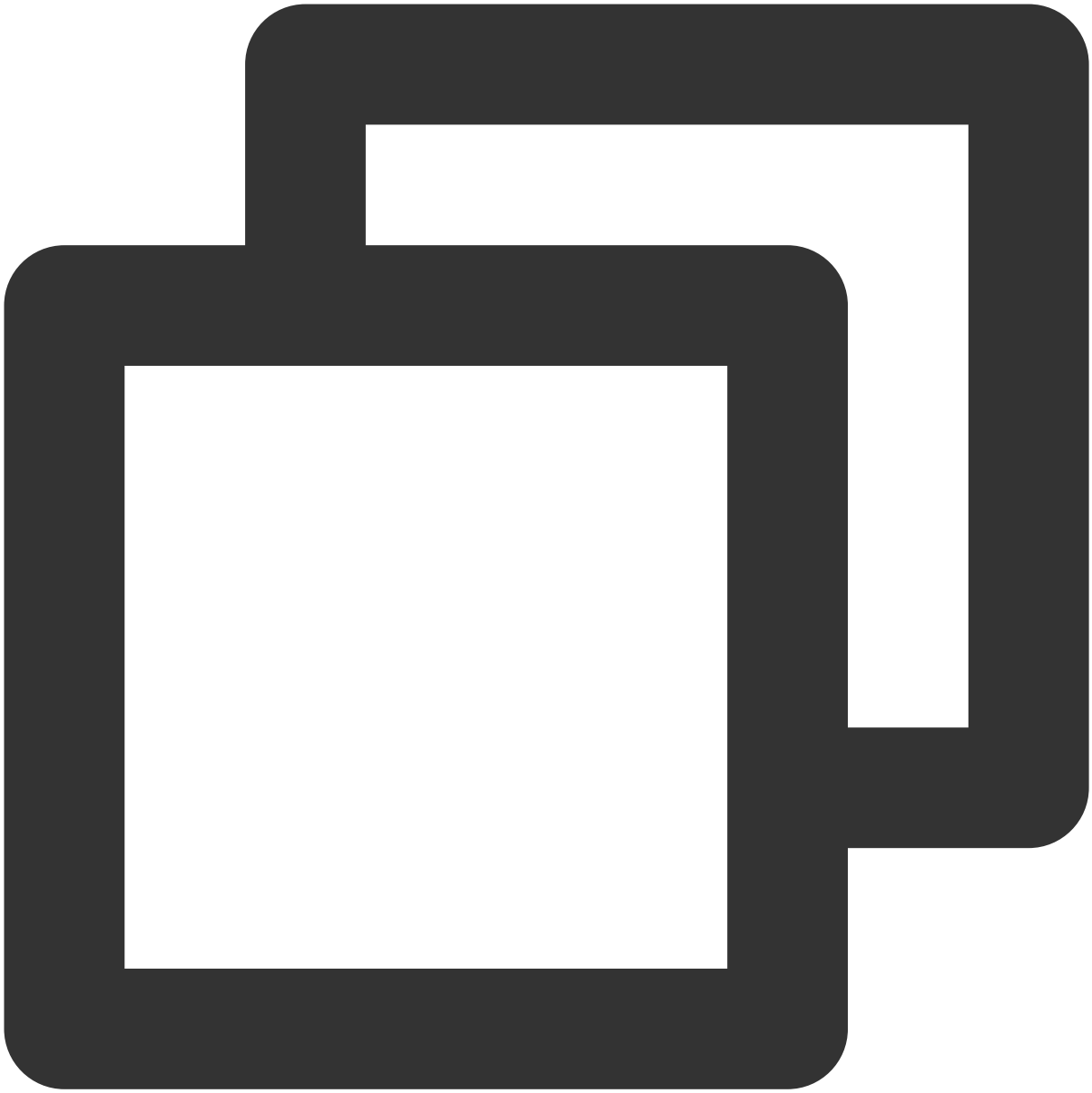
API	描述
call	发起通话。
answer	接听来电。
terminate	结束通话。
sendDTMF	发送 DTMF（双音多频信号）。
mute	静音。
unmute	取消静音。

发起呼叫和结束呼叫示例代码

Swift

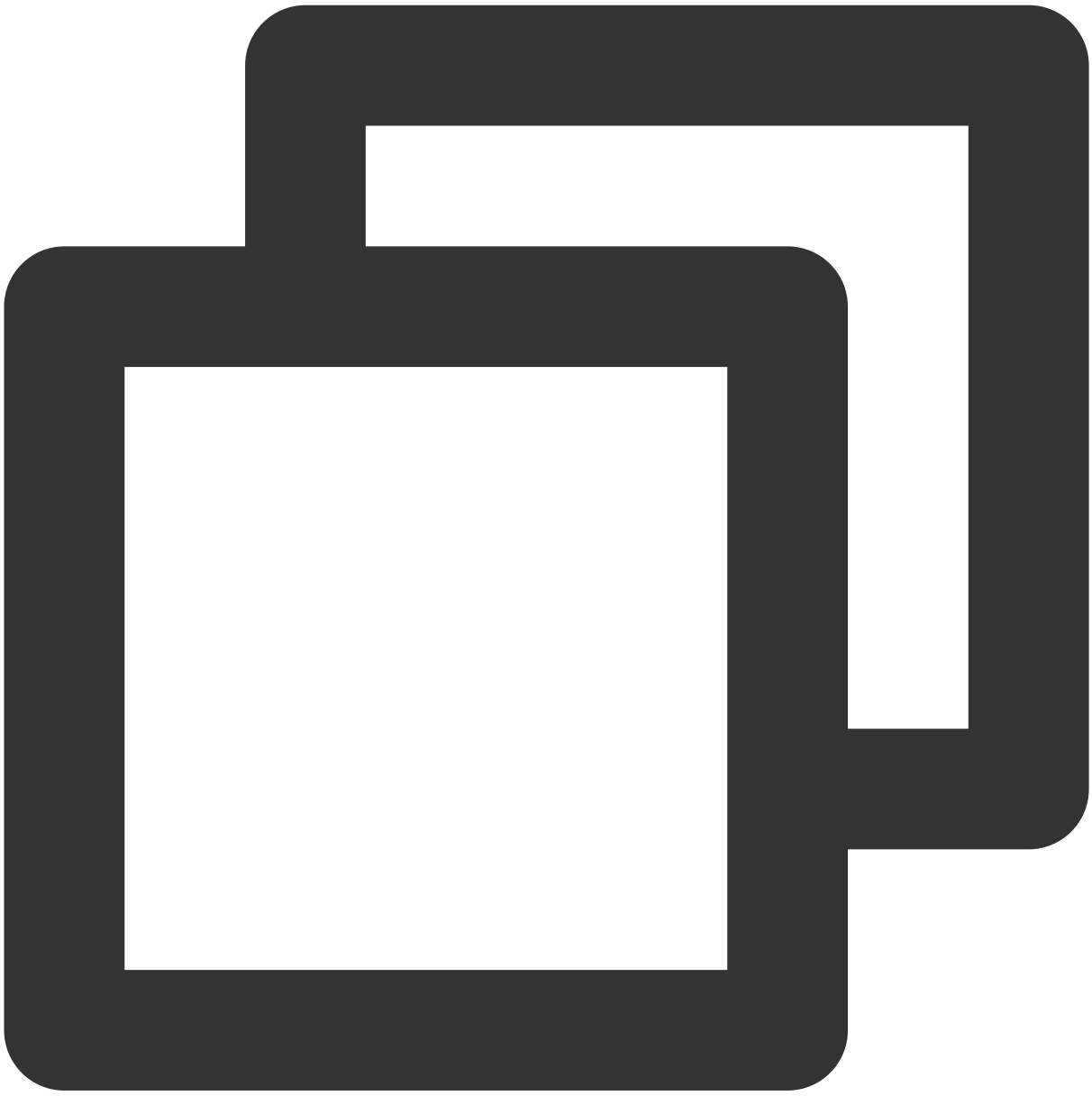
Objective-C

C++



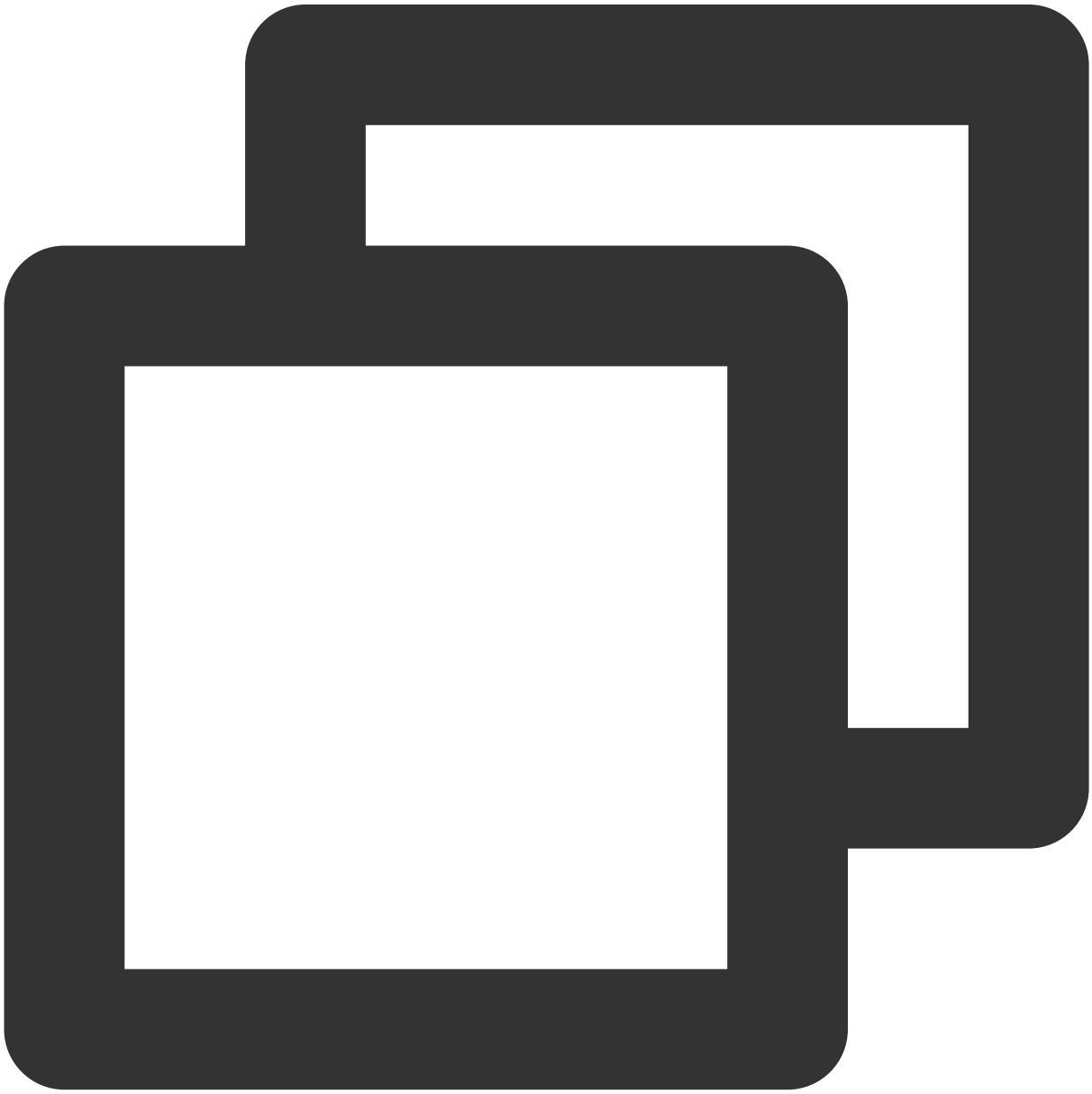
```
import TCCCSDK

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()
```



```
// 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];
```



```
#include "TCCCSDK/tccc/include/ITCCCWorkstation.h"
using namespace tccc;
class TCCCCommonCallback : public ITXCallback {
private:
    NSString* mFunName;
public:
    TCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    ~TCCCCommonCallback() 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;
//呼叫的手机号
callParams.to = "";
// 发起外呼
tcccSDK->call(callParams, startCallCallbackImpl);
// 结束通话
tcccSDK->terminate();
```

音频设备接口函数

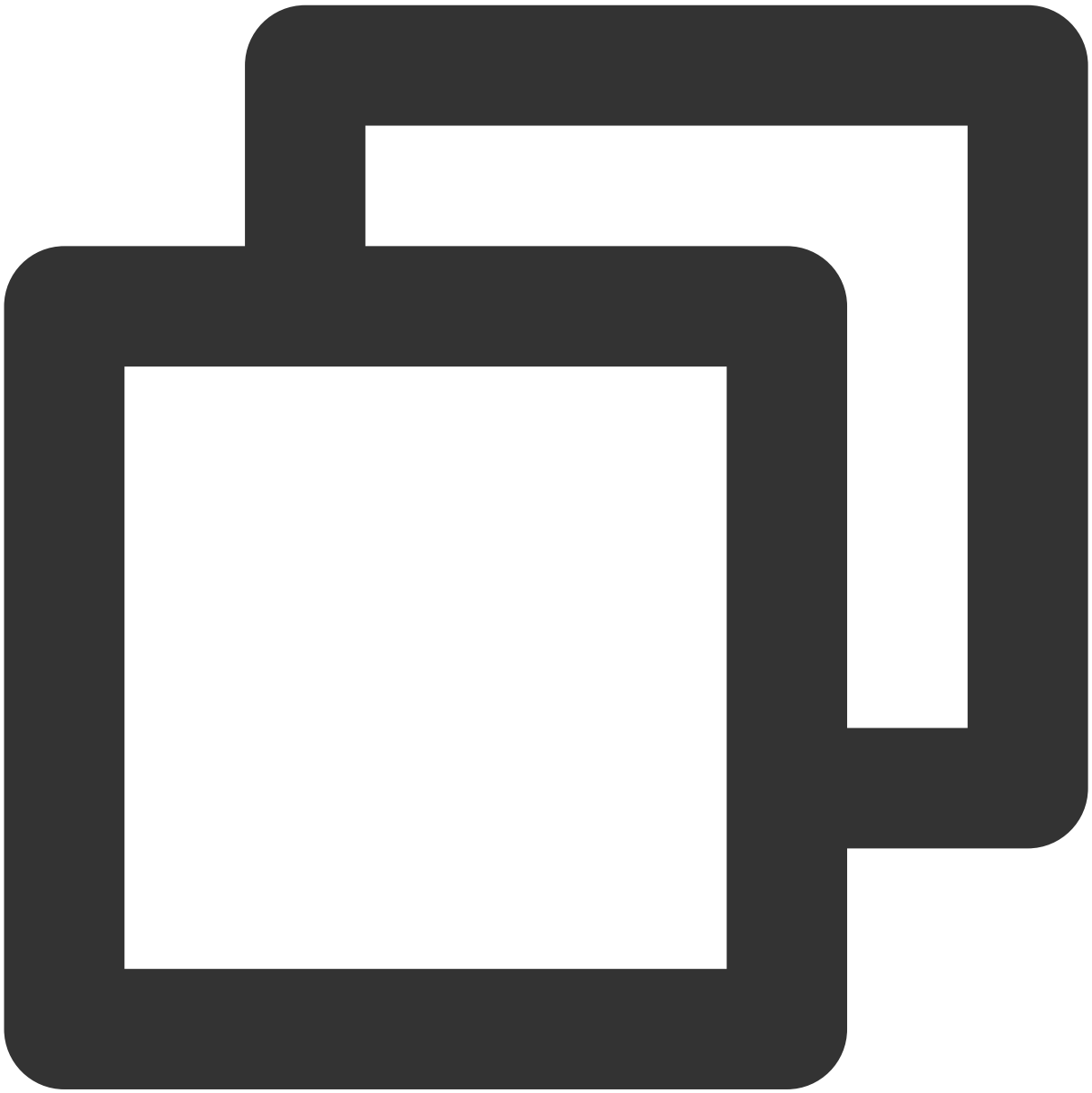
API	描述
setAudioCaptureVolume	设定本地音频的采集音量。
getAudioCaptureVolume	获取本地音频的采集音量。
setAudioPlayoutVolume	设定远端音频的播放音量。
getAudioPlayoutVolume	获取远端音频的播放音量。
setAudioRoute	设置音频路由。

切换音频路由示例代码

Swift

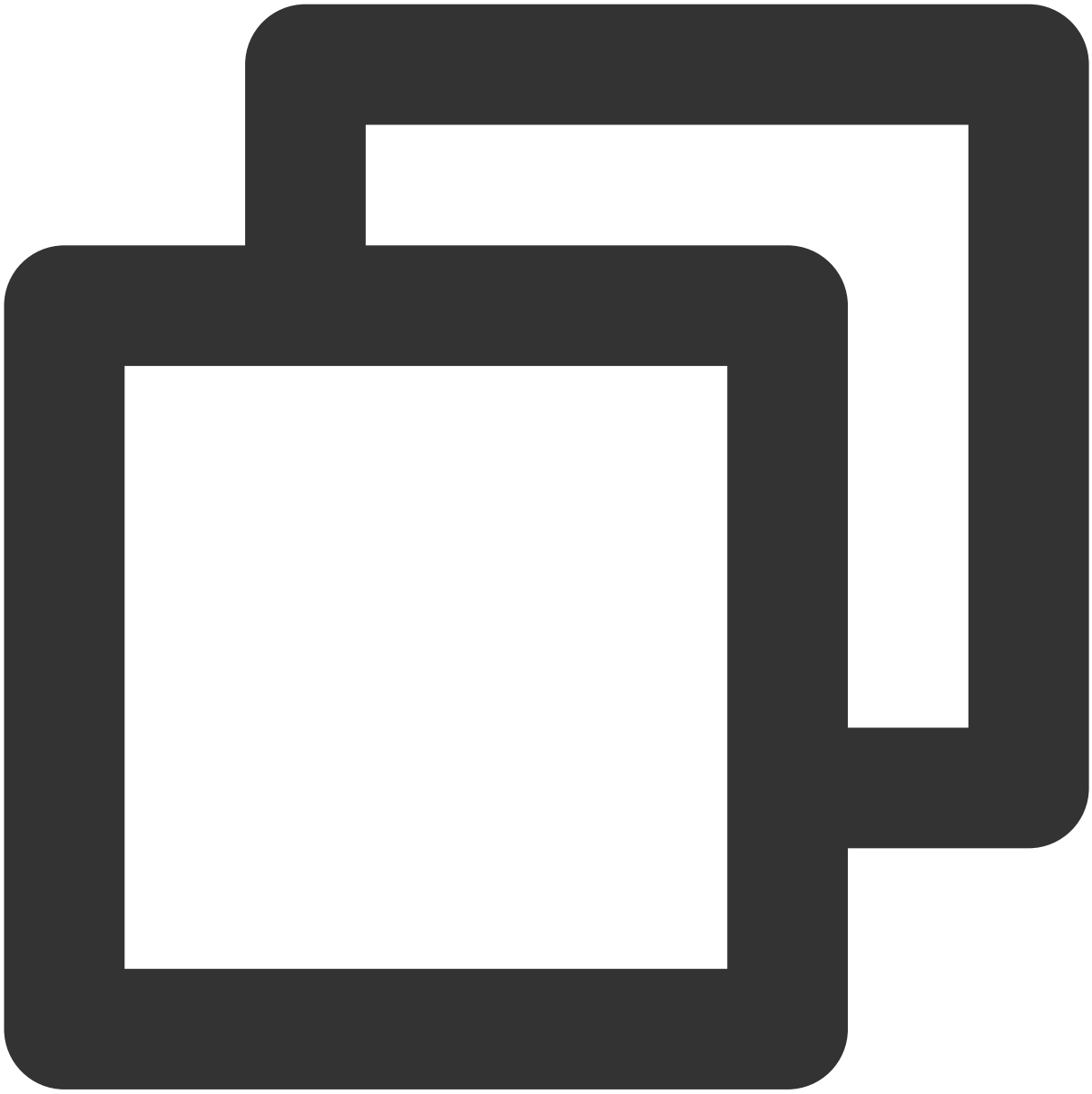
Objective-C

C++



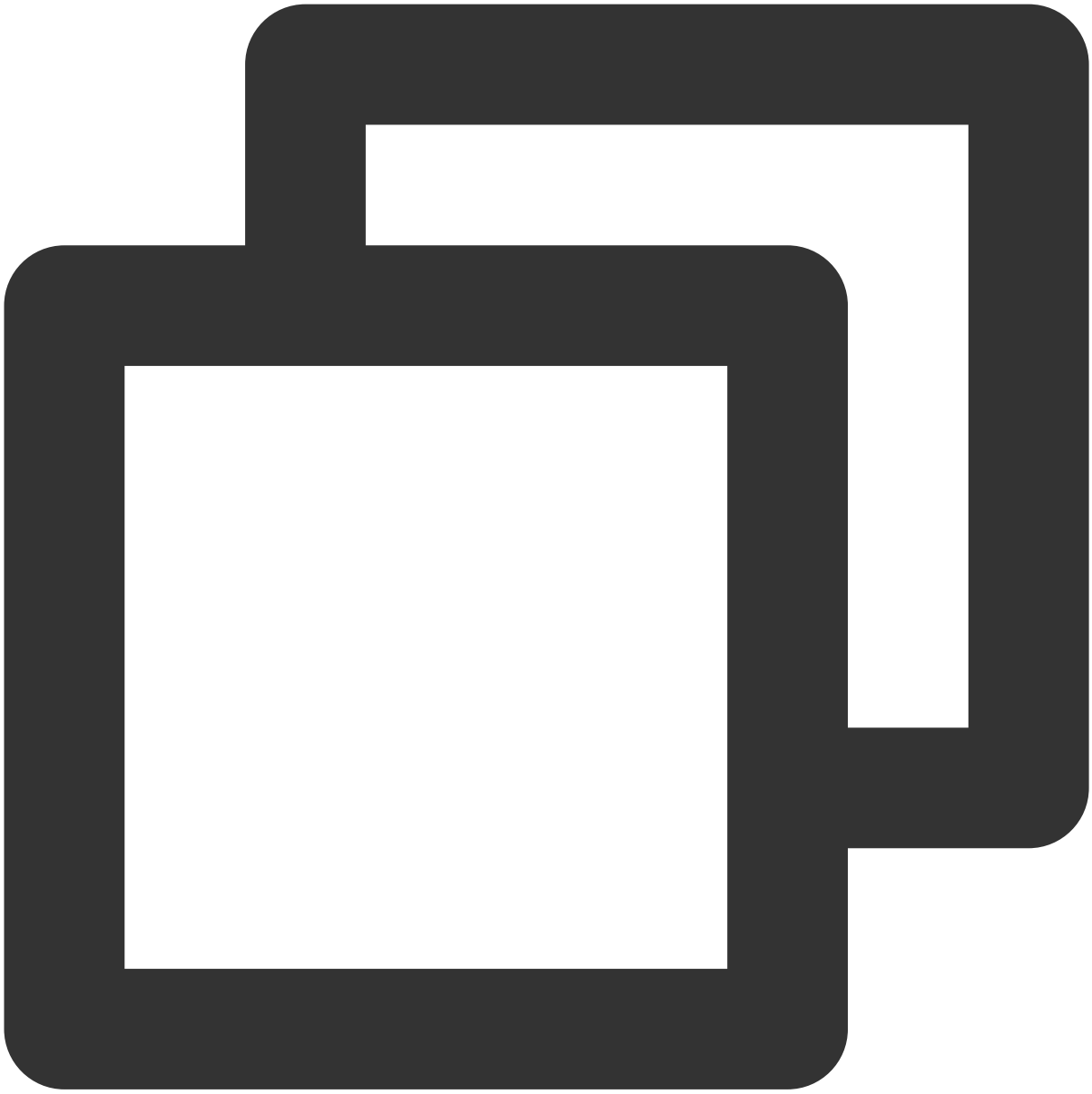
```
import TCCCSDK

// 切换为扬声器
tcccSDK.getDeviceManager().setAudioRoute(.TCCCAudioRouteSpeakerphone)
// 静音
```

```
tcccSDK.mute()
// 取消静音
tcccSDK.unmute()
```



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

// 切换为扬声器
[[self.tcccSDK getDeviceManager] setAudioRoute:TCCCAudioRouteSpeakerphone];
// 静音
[self.tcccSDK mute];
```

```
// 取消静音
[self.tcccSDK unmute];
```



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

// 切换为扬声器
tcccSDK->getDeviceManager()->setAudioRoute(TCCCAudioRoute::TCCCAudioRouteSpeakerpho
// 静音
tcccSDK->mute();
// 取消静音
```

```
tcccSDK->unmute();
```

调试相关接口

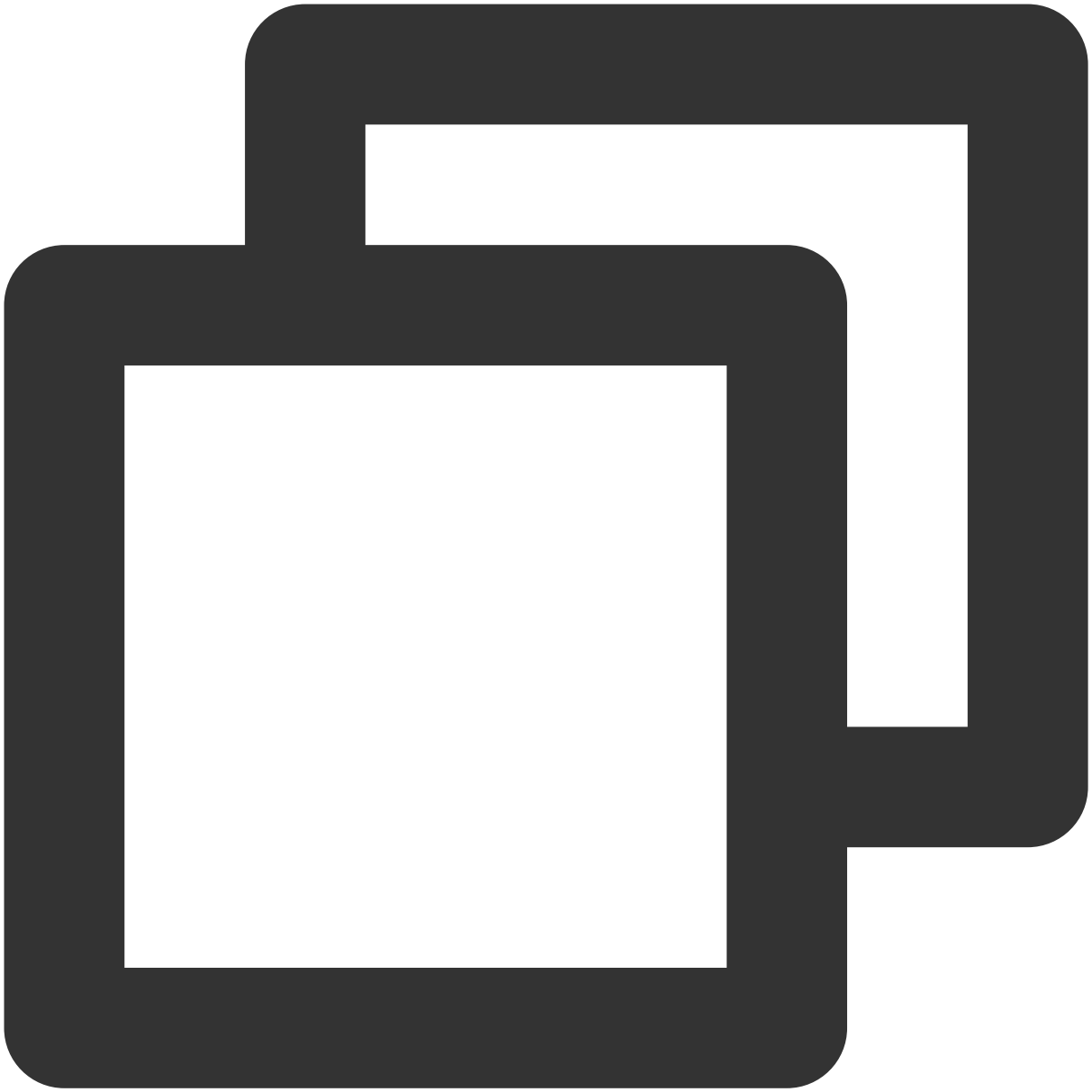
API	描述
getSDKVersion	获取 SDK 版本信息。
setLogLevel	设置 Log 输出级别。
setConsoleEnabled	启用/禁用控制台日志打印。
callExperimentalAPI	调用实验性接口。

获取SDK版本示例代码

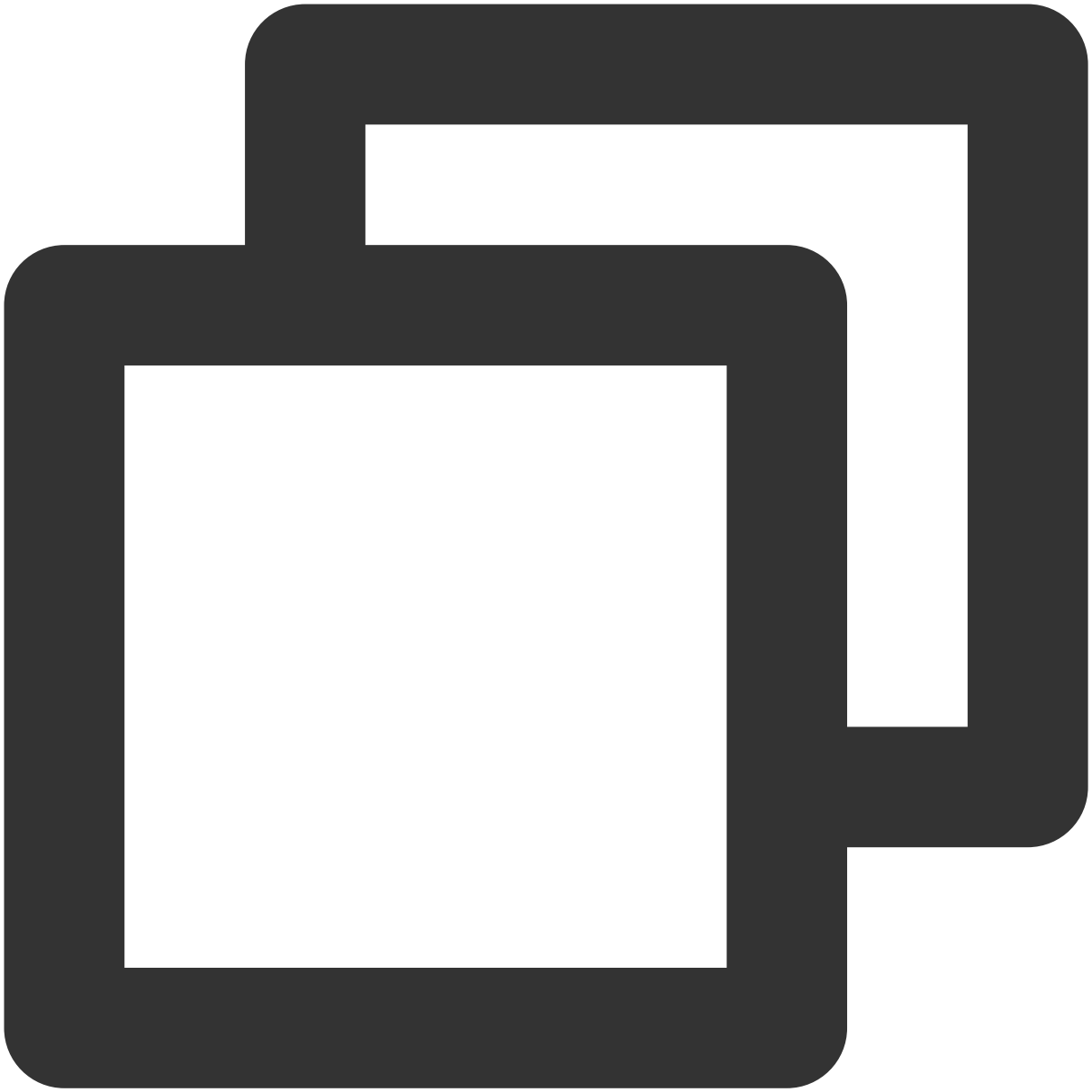
Swift

Objective-C

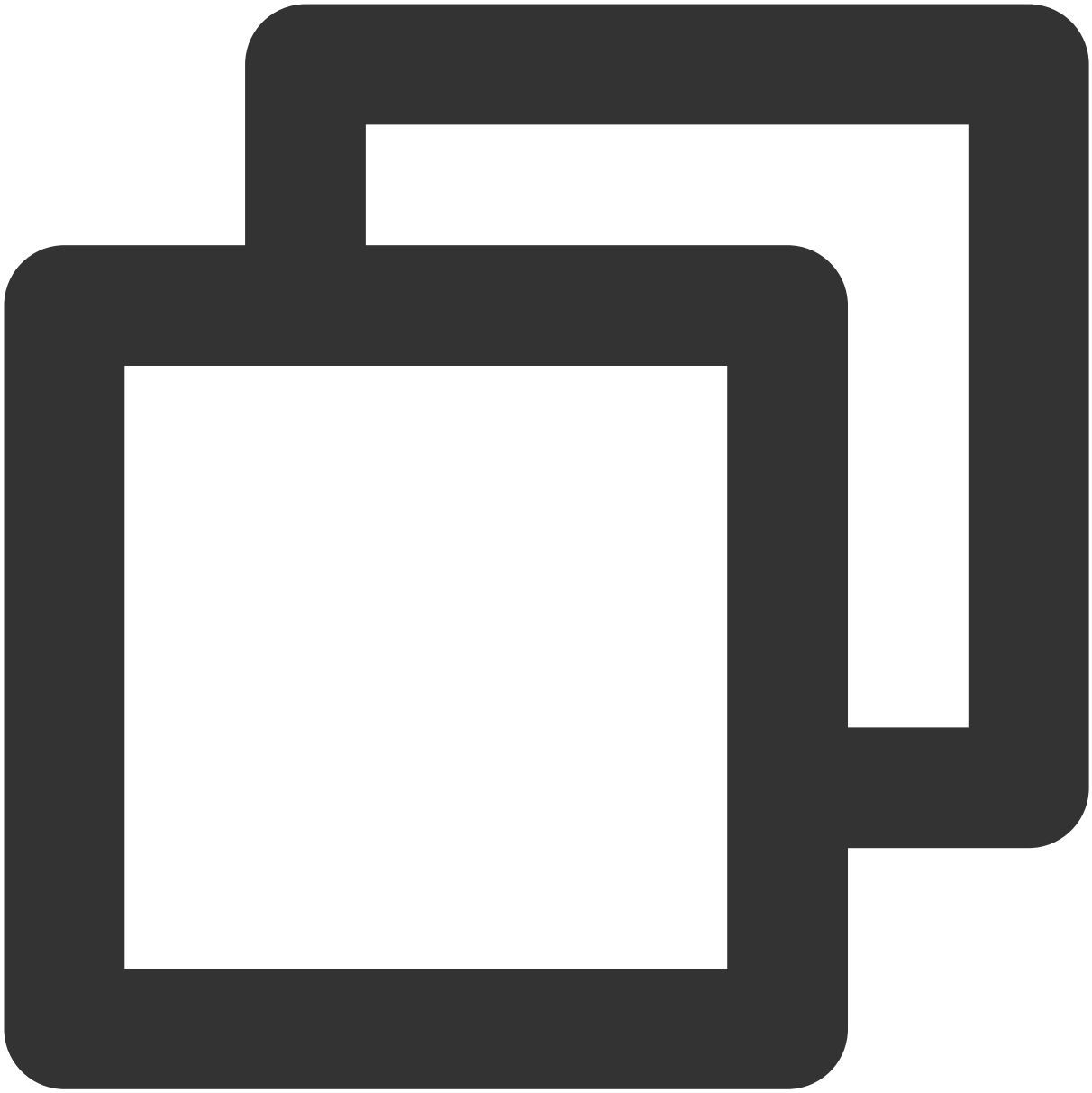
C++



```
import TCCCSDK

// 获取SDK版本号
let version = TCCCWorkstation.getSDKVersion()
```



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

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



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

// 获取SDK 版本号
tcccSDK->getSDKVersion();
```

错误和警告事件

API	描述

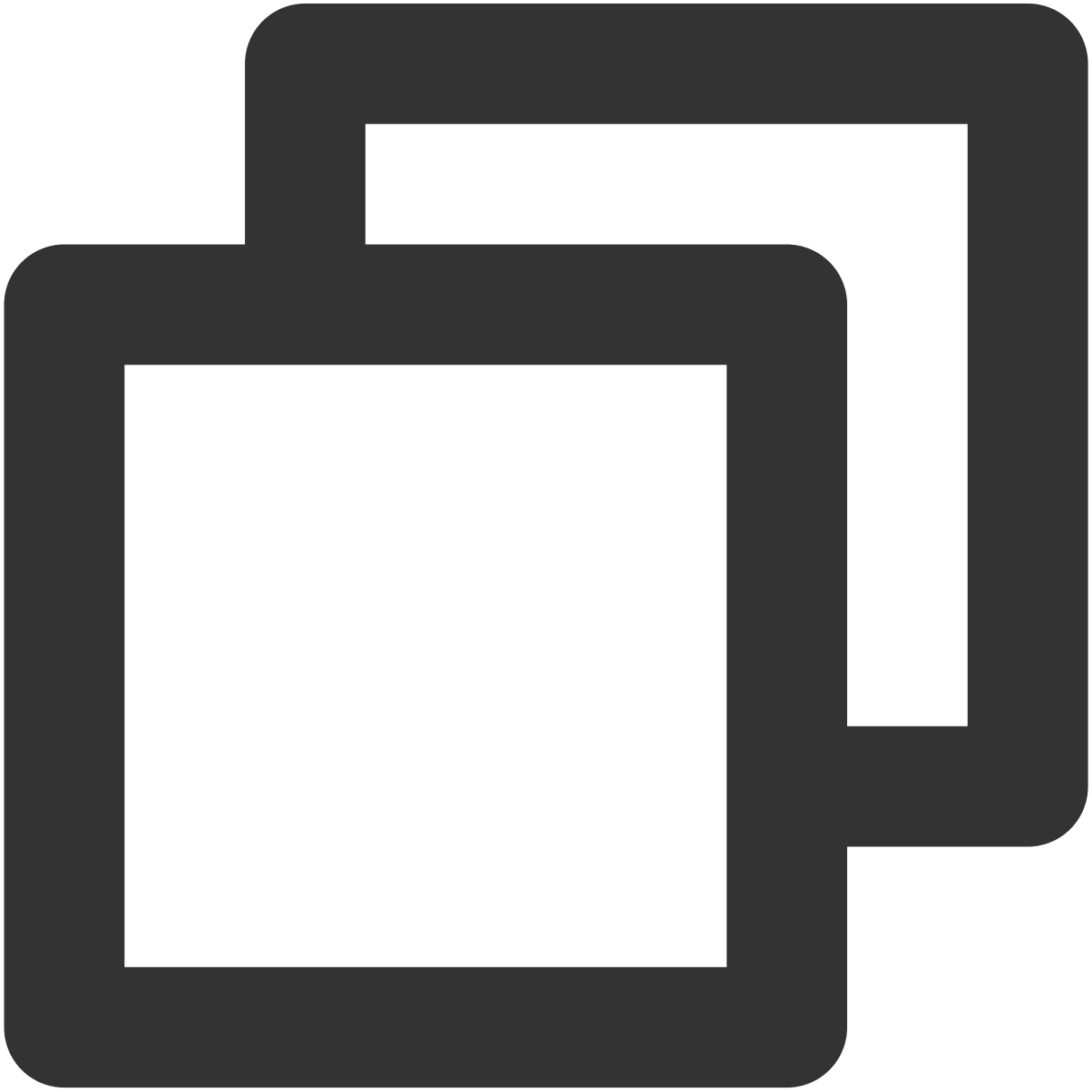
onError	错误事件回调。
onWarning	警告事件回调。

处理错误回调事件回调示例代码

Swift

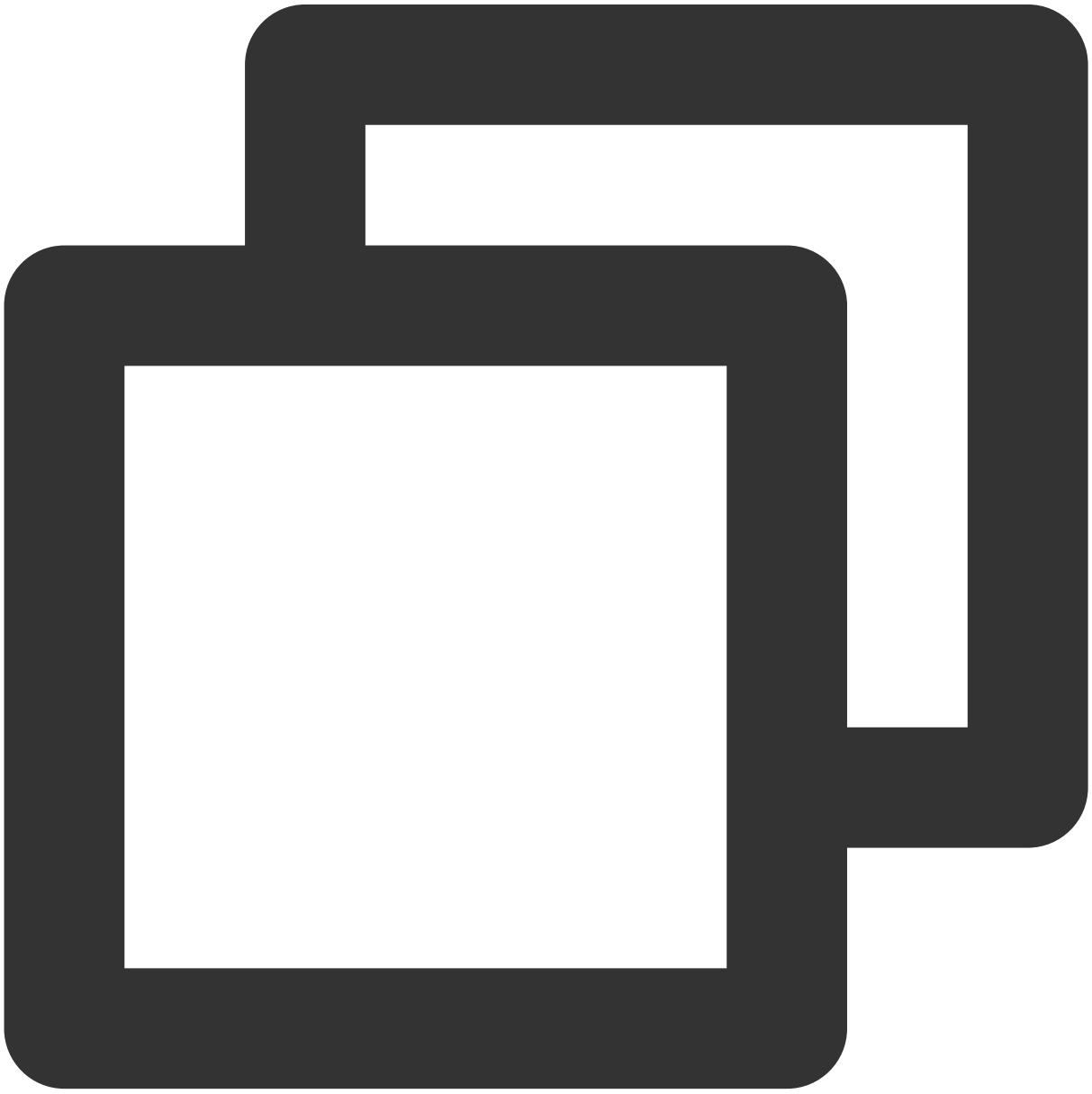
Objective-C

C++

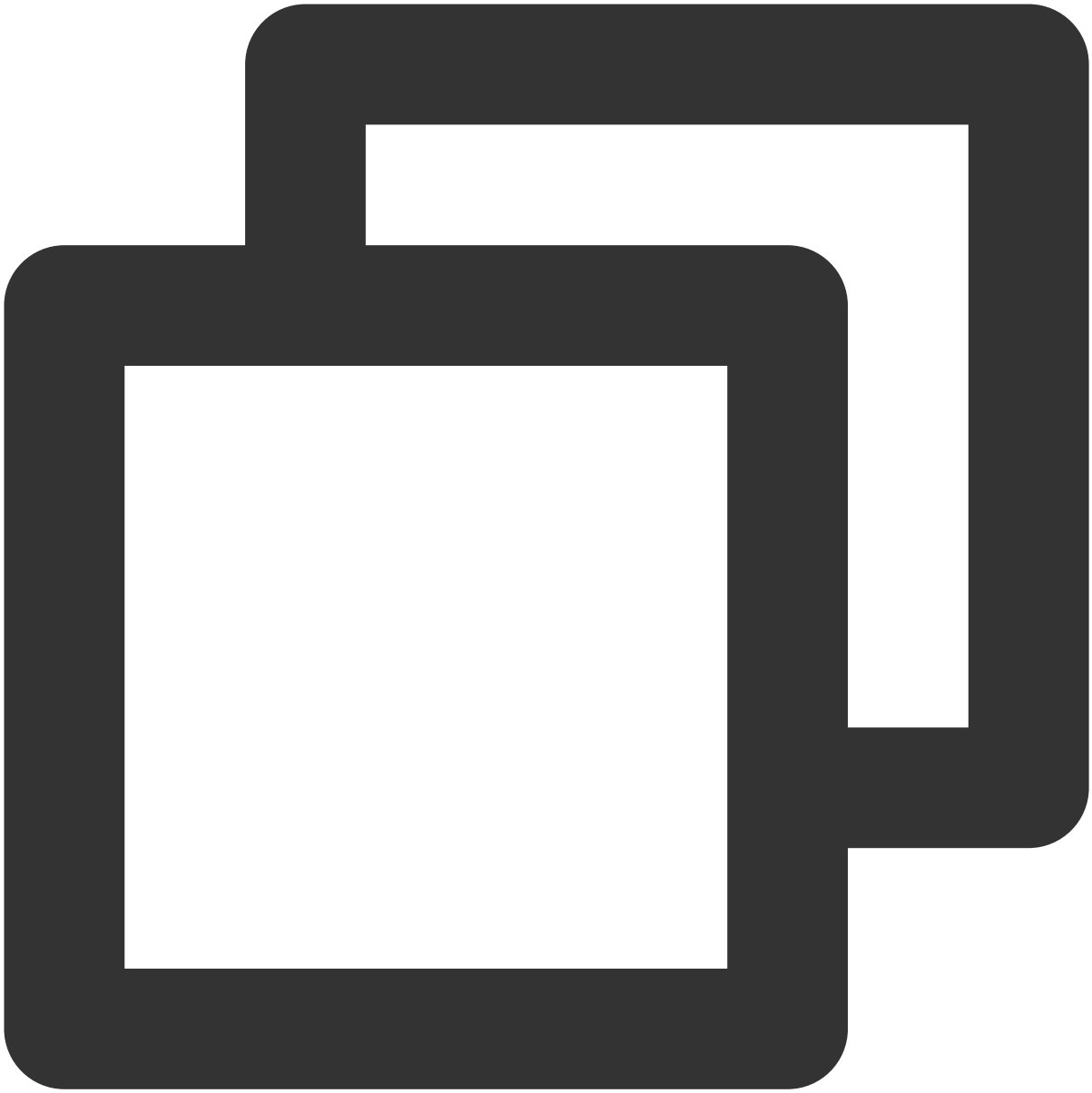


```
import TCCCSDK
```

```
func onError(_ errCode: TCCCErrorCode, errMsg: String, extInfo: [AnyHashable : Any]
    // 错误事件回调
}
func onWarning(_ warningCode: TCCCCWarningCode, warningMsg: String, extInfo: [AnyHa
    // 警告事件回调
}

// 设置TCCC事件回调
tcccSDK.addTcccListener(self)
```



```
// 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:(
    // 错误事件回调
}
- (void)onWarning:(TCCCCWarningCode)warningCode warningMsg:(NSString *_Nonnull)warn
    // 警告事件回调
}

// 设置TCCC事件回调
[self.tcccSDK addTcccListener:self];
```



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

// 设置回调，TCCCCallbackImpl 需要继承 ITCCCCallback
class TCCCCallbackImpl:public ITCCCCallback {
public:
    TCCCCallbackImpl() {}
    ~TCCCCallbackImpl() {}
    // 错误事件回调
    void onError(TCCCError errCode, const char* errMsg, void* extraInfo) {}
    // 警告事件回调
```

```
    void onWarning(TCCCCWarning warningCode, const char* warningMsg, void* extraInf
};
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();
tcccSDK->addCallback(tcccCallback);
```

呼叫相关事件回调

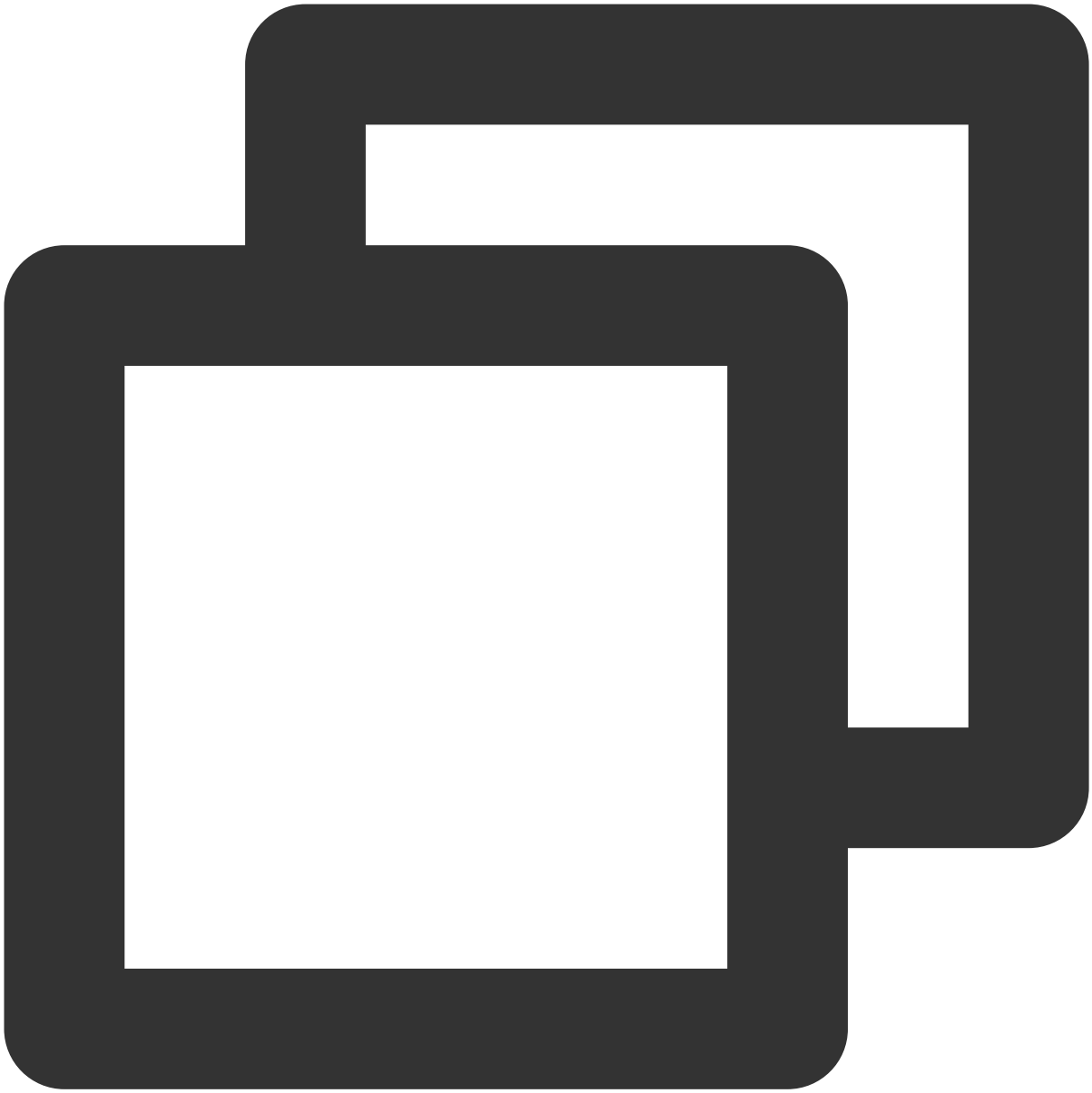
API	描述
onNewSession	新会话事件。包括呼入和呼出。
onEnded	会话结束事件。
onAudioVolume	音量大小的反馈回调。
onNetworkQuality	网络质量的实时统计回调。

处理接听和坐席挂断事件回调示例代码

Swift

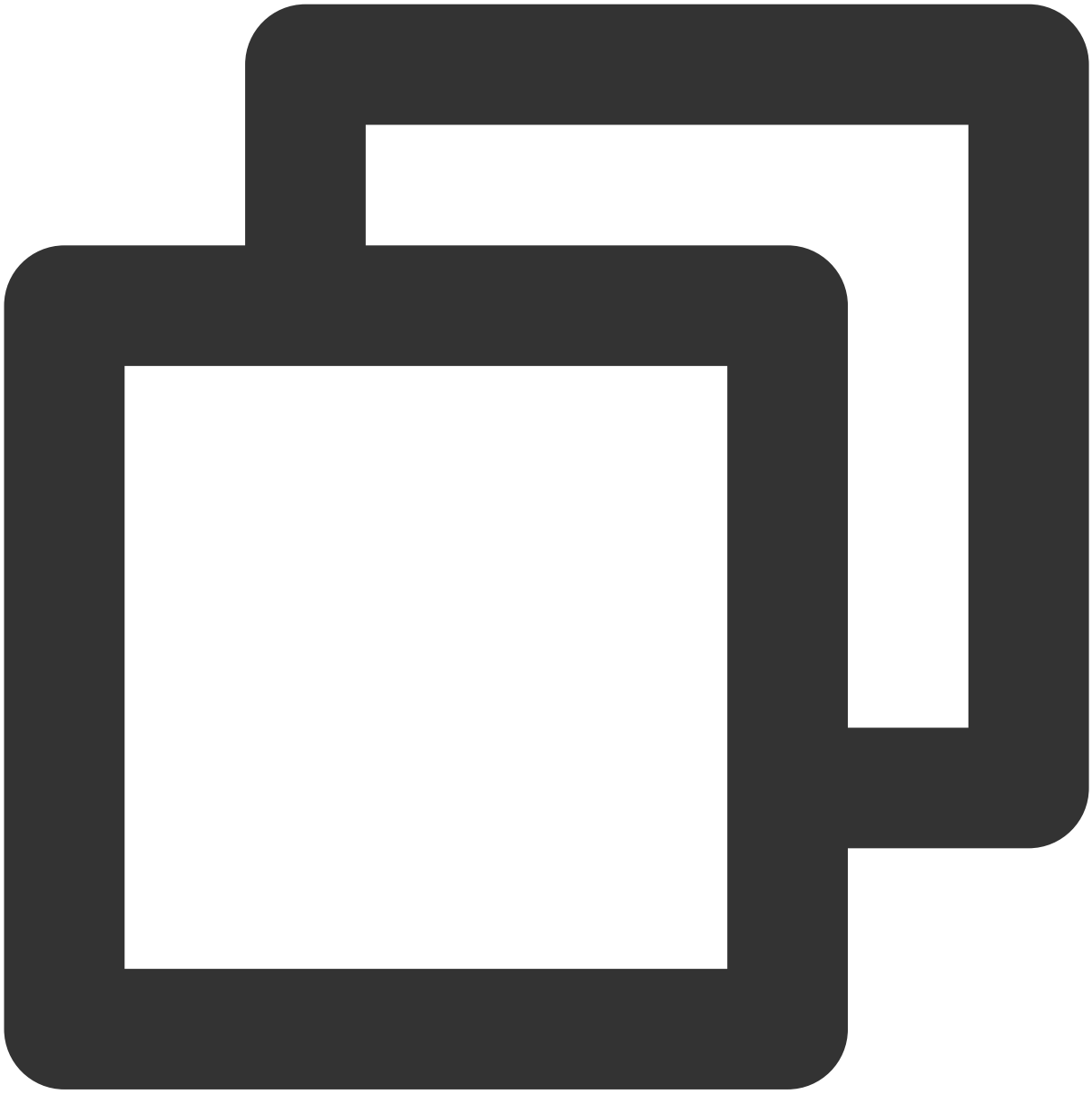
Objective-C

C++



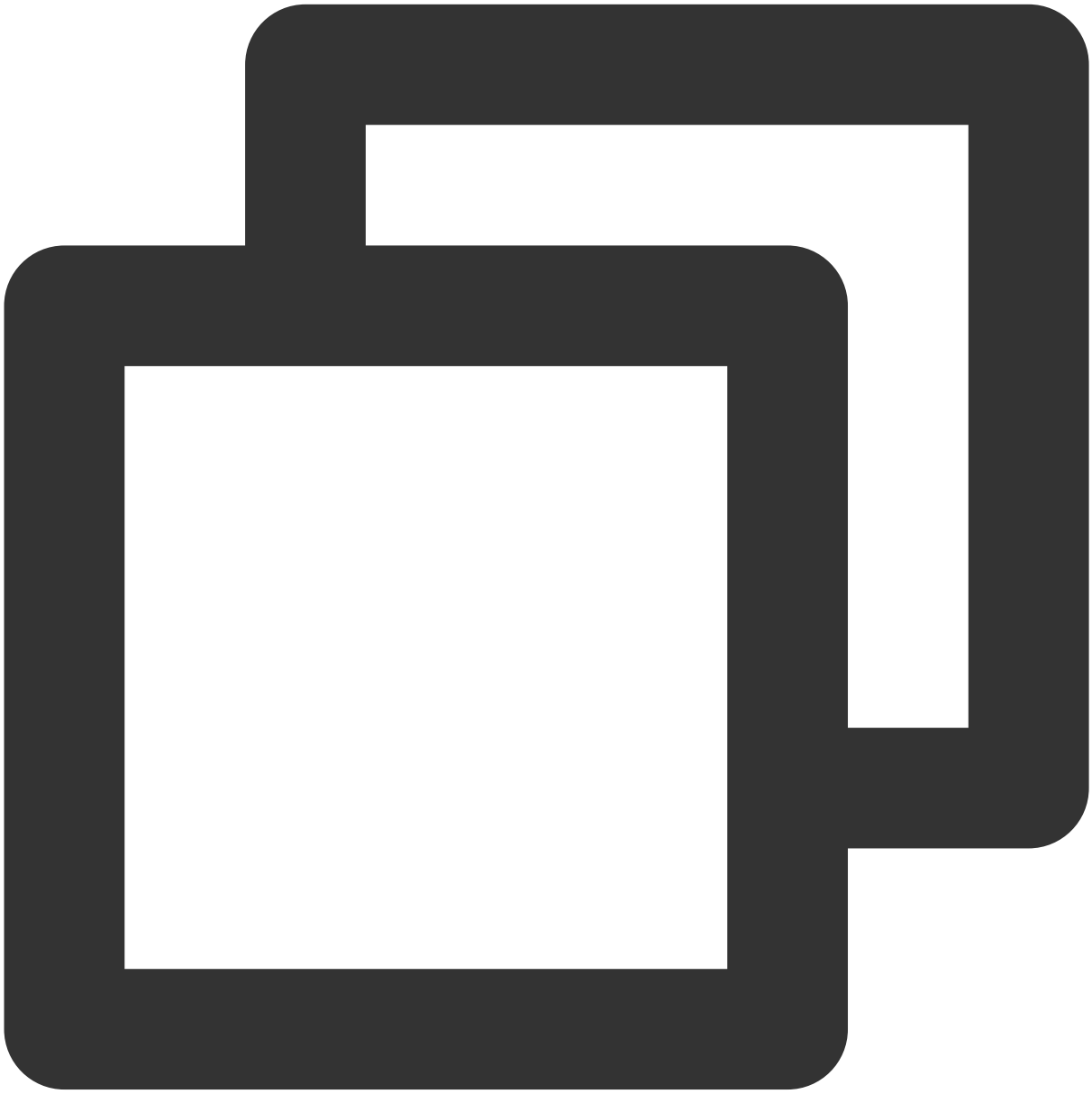
```
import TCCCSDK

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)
```



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

- (void)onNewSession:(TXSessionInfo *)info {
    // 新会话事件。包括呼入和呼出
}
- (void)onEnded:(TXEndedReason)reason reasonMessage:(NSString *_Nonnull)reasonMessa
```

```
    // 通话结束事件
}
- (void)onAccepted:(NSString *_Nonnull)sessionId {
    // 对端已接听事件
}
// 设置TCCC事件回调
[self.tcccSDK addTcccListener:self];
```



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

```
// 设置回调，TCCCCallbackImpl 需要继承 ITCCCCallback
class TCCCCallbackImpl:public ITCCCCallback {
public:
    TCCCCallbackImpl() {}
    ~TCCCCallbackImpl() {}
    // 新会话事件。包括呼入和呼出
    void onNewSession(TCCCSessionInfo info) {}
    // 会话结束事件
    void onEnded(EndedReason reason, const char* reasonMessage, const char* session

};
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();
tcccSDK->addCallback(tcccCallback);
```

与云端连接情况的事件回调

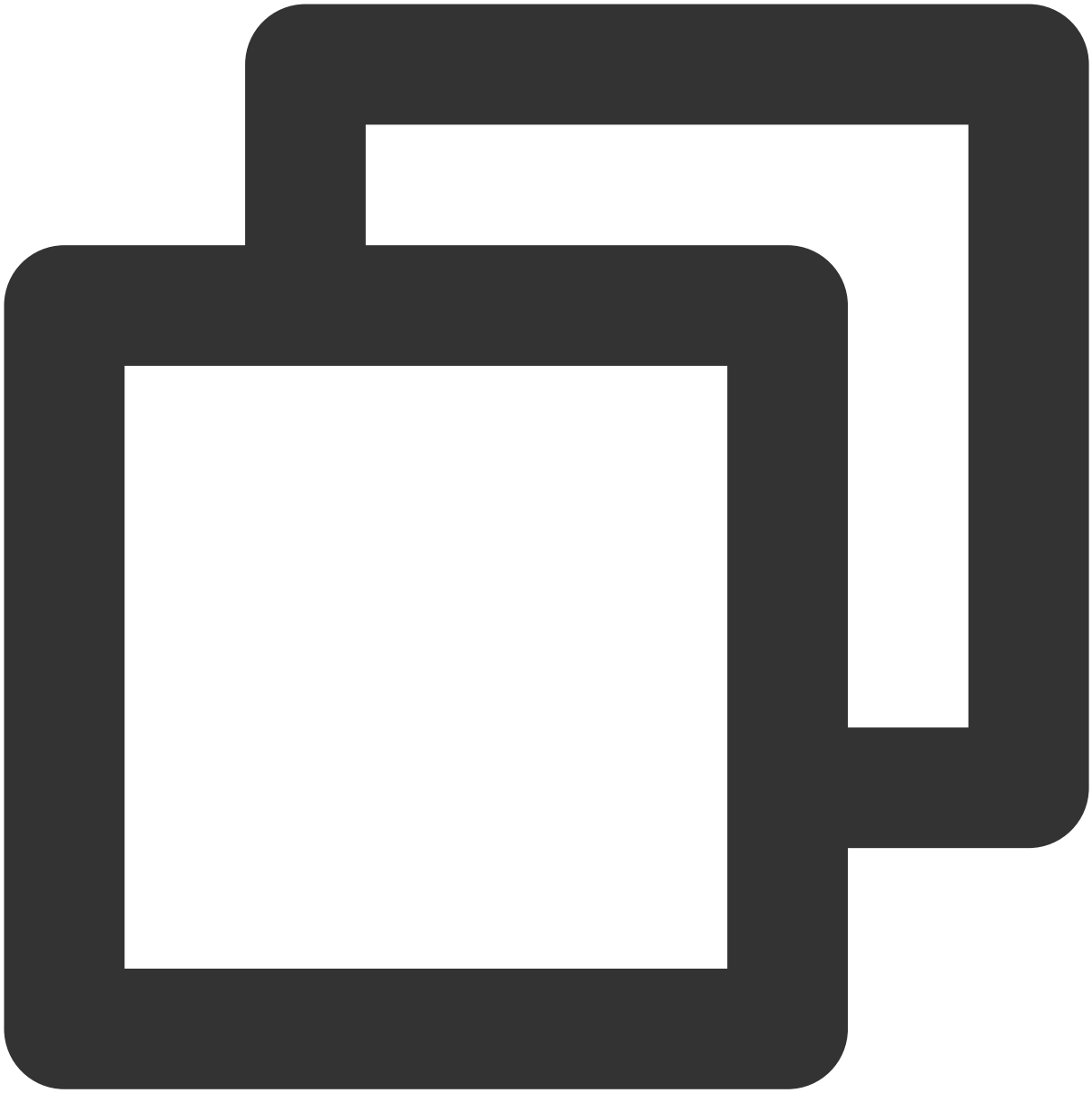
API	描述
onConnectionLost	SDK 与云端的连接已经断开。
onTryToReconnect	SDK 正在尝试重新连接到云端。
onConnectionRecovery	SDK 与云端的连接已经恢复。

与云端连接情况的事件回调示例代码

Swift

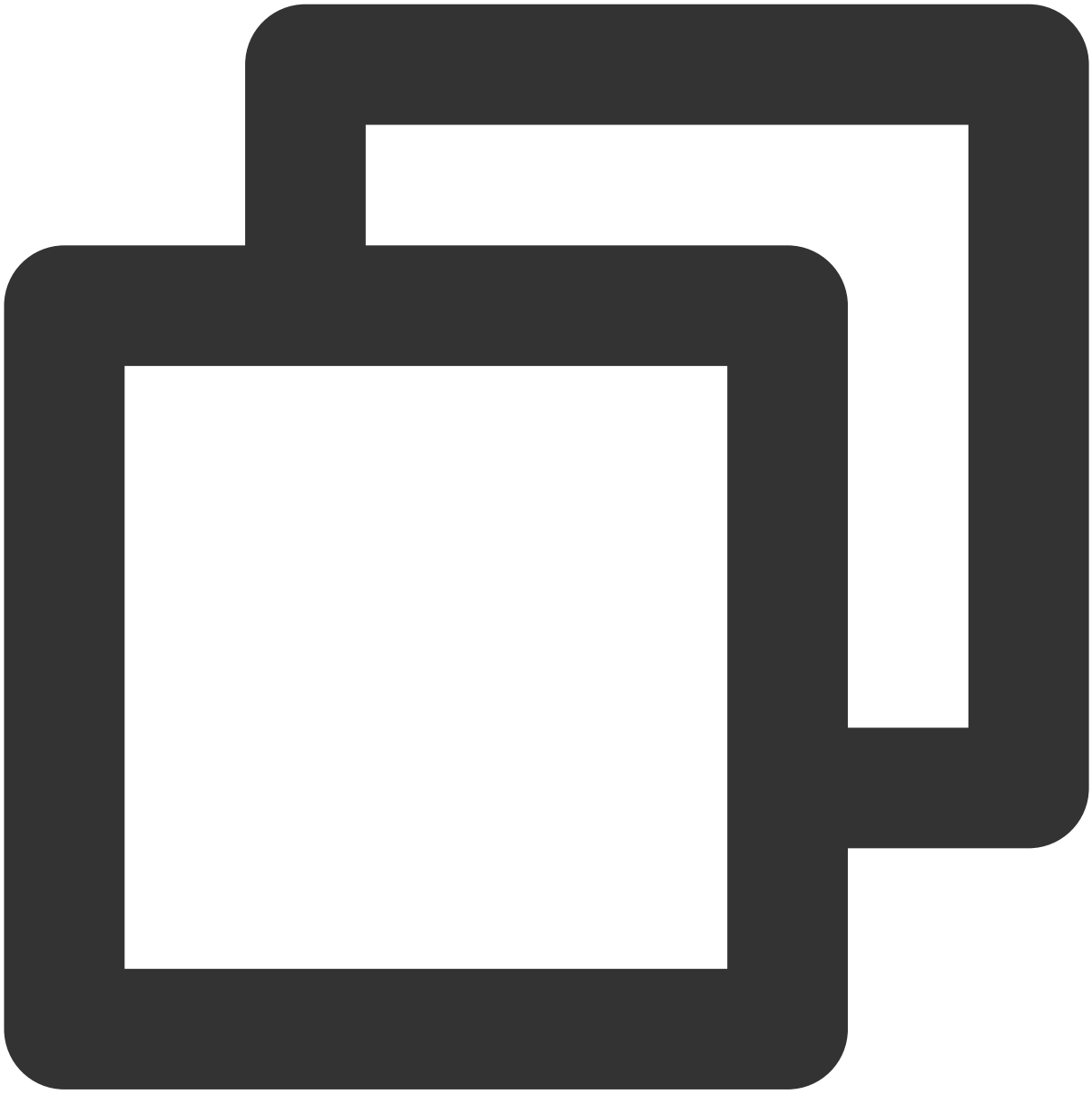
Objective-C

C++



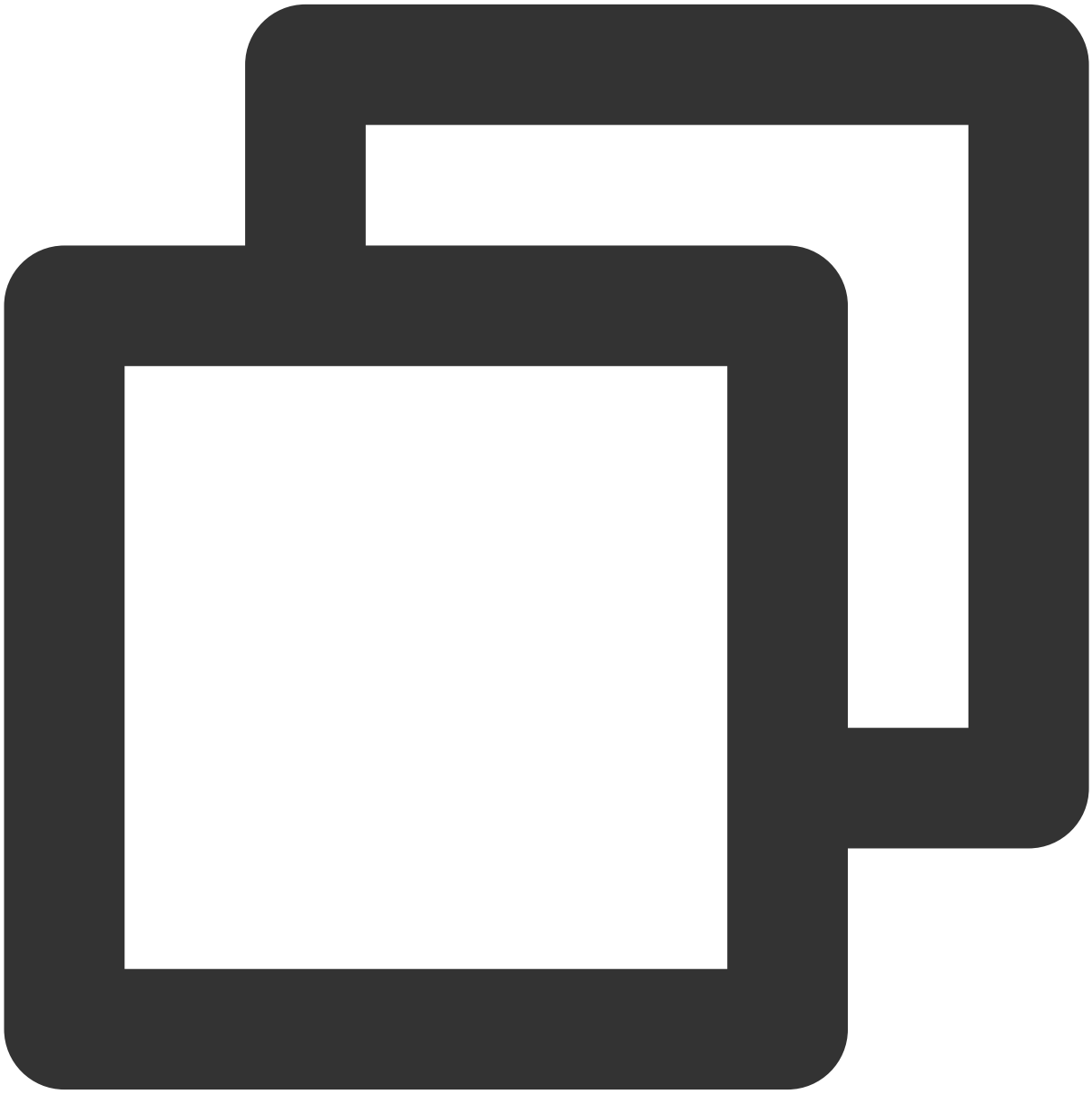
```
import TCCCSDK

func onConnectionLost(_ serverType: TXServerType) {
    // SDK 与云端的连接已经断开
}
func onConnectionRecovery(_ serverType: TXServerType) {
    // SDK 与云端的连接已经恢复
}
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)
```



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

- (void)onConnectionLost:(TXServerType)serverType {
    // SDK 与云端的连接已经断开
}
- (void)onTryToReconnect:(TXServerType)serverType {
```

```
    // SDK 正在尝试重新连接到云端
}
- (void)onConnectionRecovery:(TXServerType)serverType {
    // SDK 与云端的连接已经恢复
}
// 设置TCCC事件回调
[self.tcccSDK addTcccListener:self];
```



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

```
// 设置回调，TCCCCallbackImpl 需要继承 ITCCCCallback
class TCCCCallbackImpl:public ITCCCCallback {
public:
    TCCCCallbackImpl() {}
    ~TCCCCallbackImpl() {}
    // SDK 与云端的连接已经断开
    void onConnectionLost(TCCCServerType serverType) {}
    // SDK 正在尝试重新连接到云端
    void onTryToReconnect(TCCCServerType serverType) {}
    // SDK 与云端的连接已经恢复
    void onConnectionRecovery(TCCCServerType serverType) {}
};
TCCCCallbackImpl* tcccCallback = new TCCCCallbackImpl();
tcccSDK->addCallback(tcccCallback);
```

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 联系我们。

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_AUTHENTICATION_REQUIRED	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_SAMPLERATE	-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_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_REFUSED	-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。

常见问题

Web SDK 常见问题

最近更新时间：2024-04-01 17:54:48

Web SDK 支持什么框架？

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，UserId 是什么？

UserId 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UserId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：2024-04-01 17:56:50

如何查看 TCCC 日志？

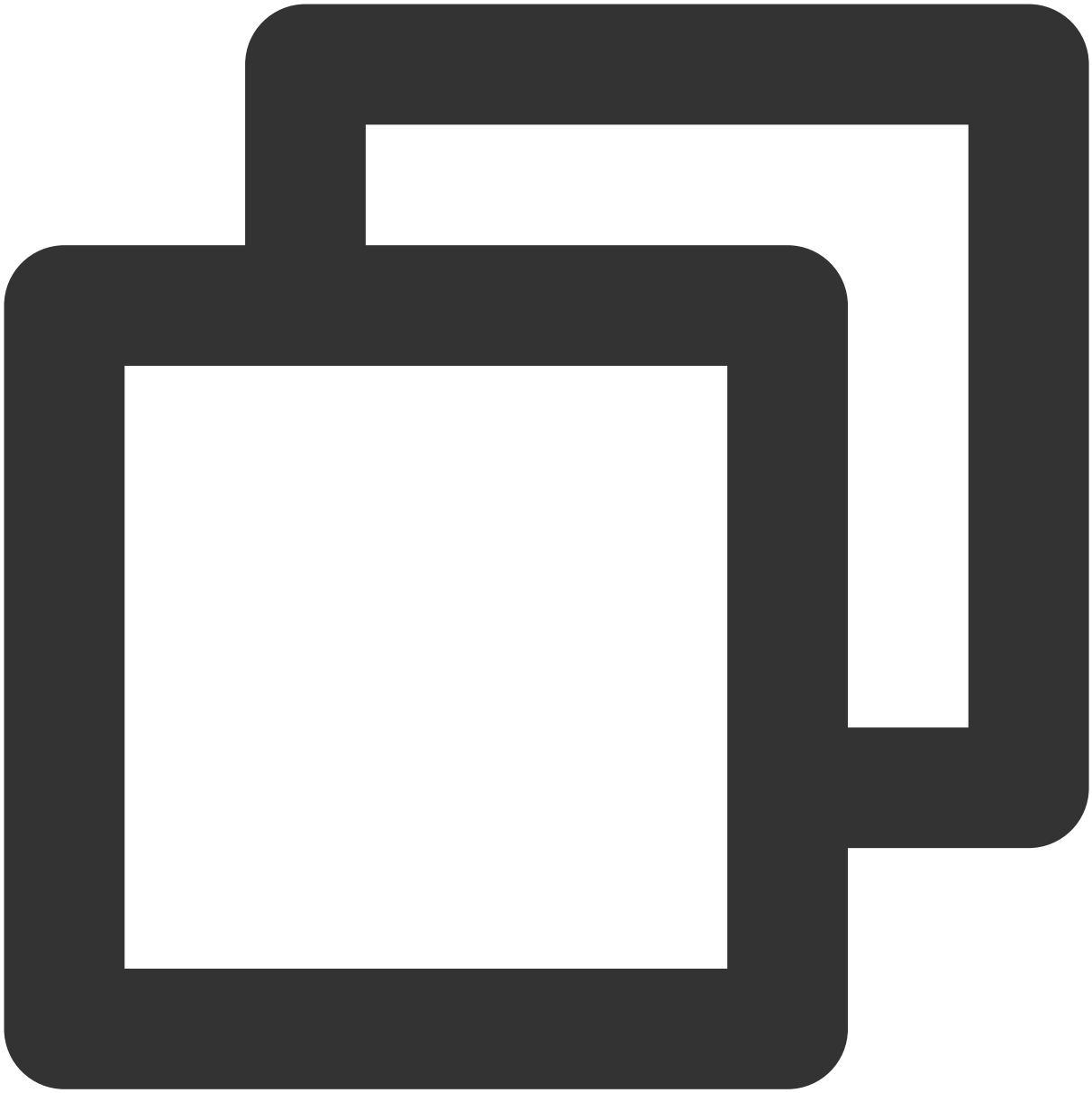
TCCC 的日志默认压缩加密，后缀为 .log。
日志路径：
Android： `/sdcard/Android/data/包名/files/tccc`
iOS：在 **sandbox/Documents** 目录下的 **tccc** 文件夹

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：2024-04-01 17:59:03

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？

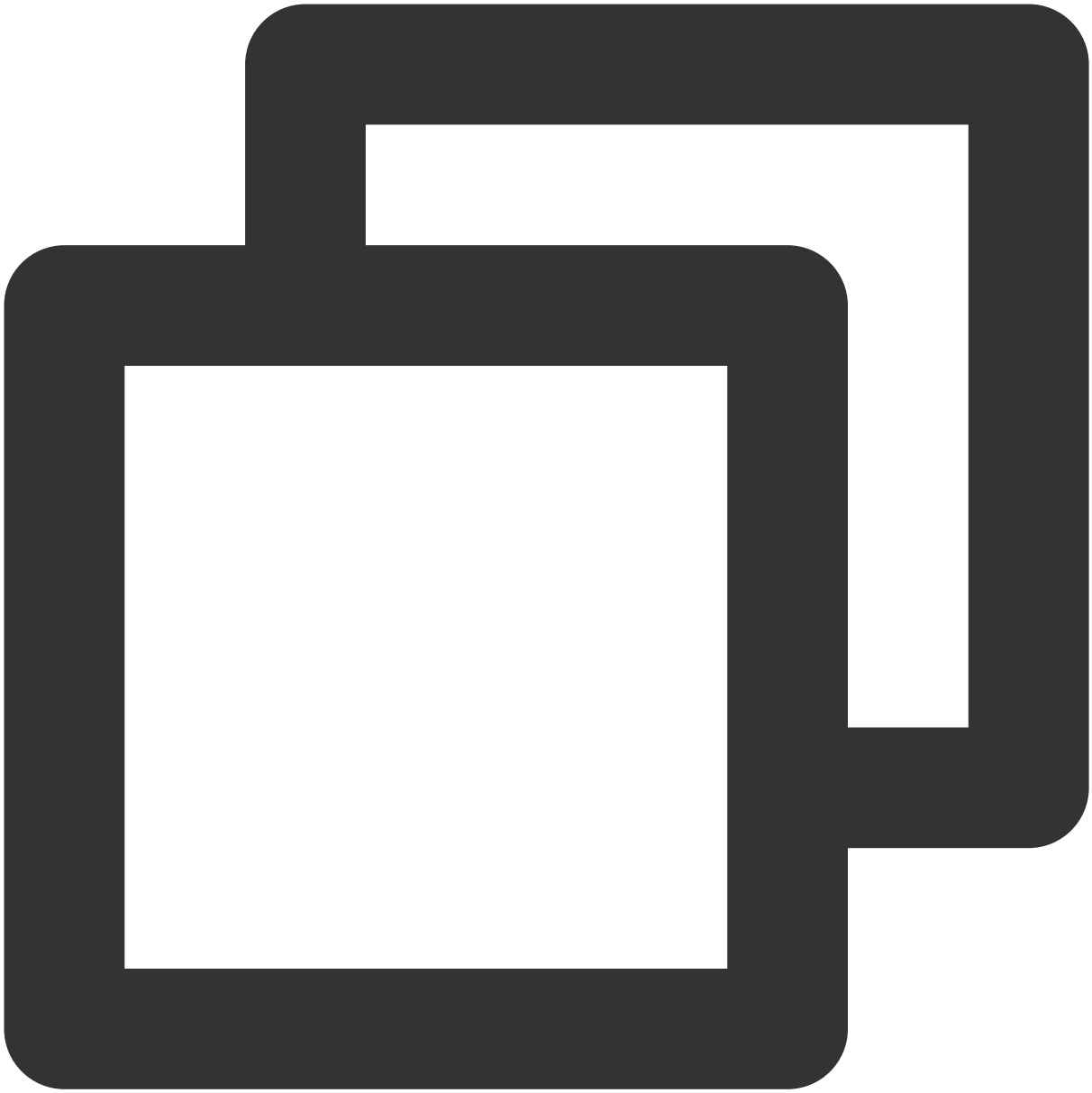
TCCC 目前版本暂时不支持，未来会支持模拟器。

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，或者设置保持屏幕常亮来解决。

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。
});
```

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。

集成电话客服
实现一键外呼
Web

最近更新时间：2024-04-01 18:08:50

步骤一：初始化 SDK

请参见 初始化SDK

注意

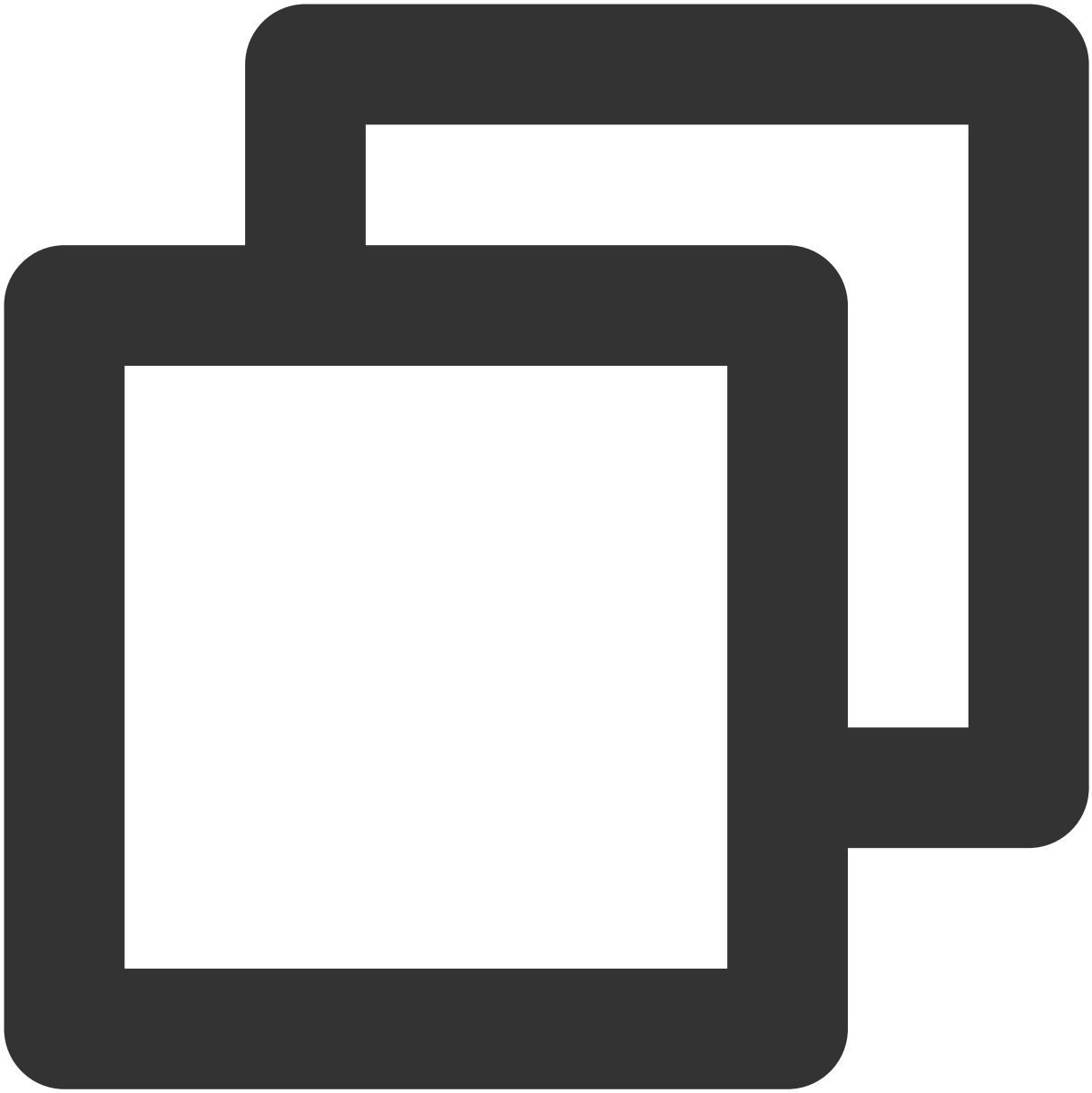
后续步骤需要在 tccc.events.ready 事件成功后才能执行。

步骤二：实现点击按钮触发SDK外呼

Vue

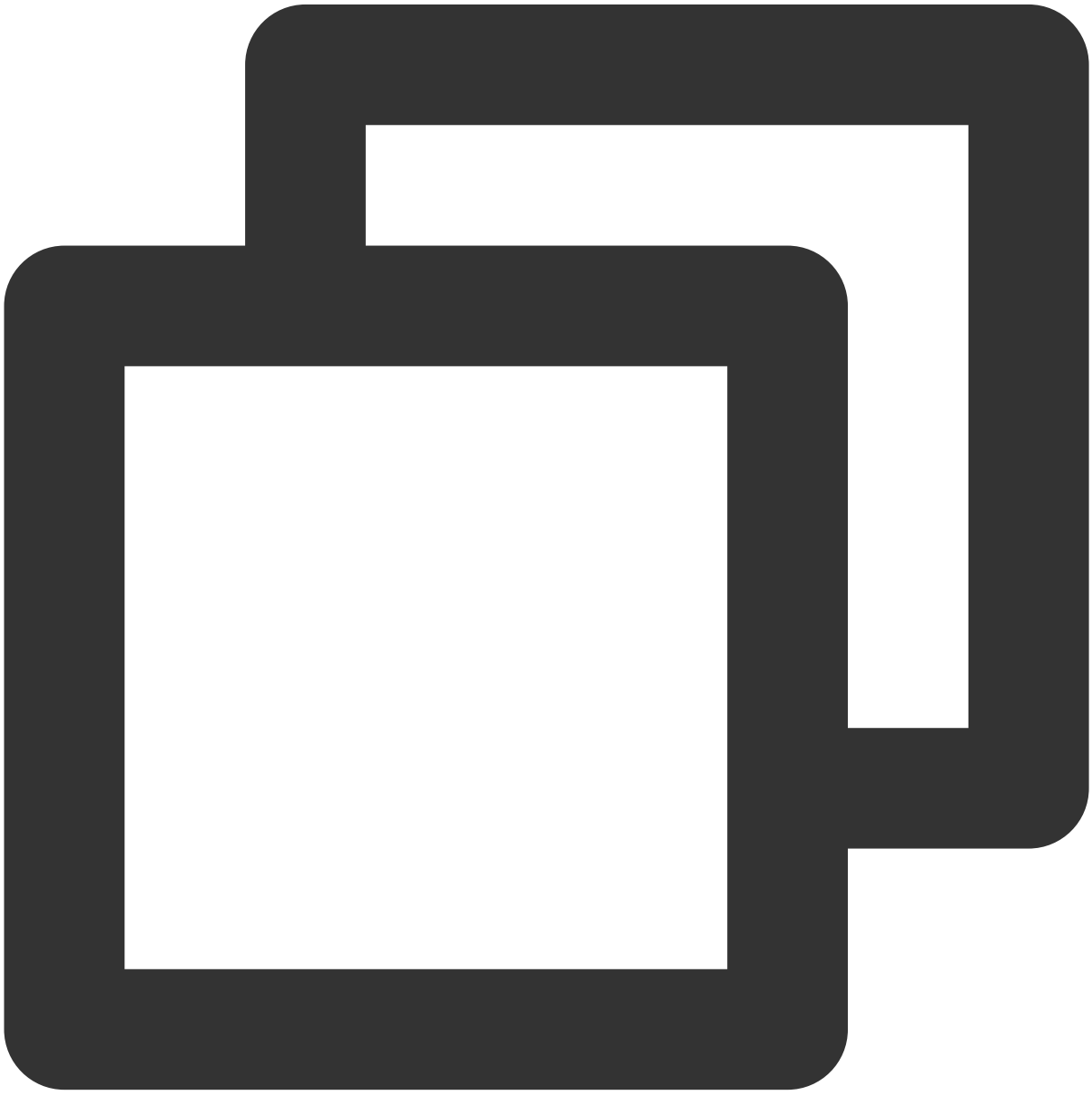
React

原生JS

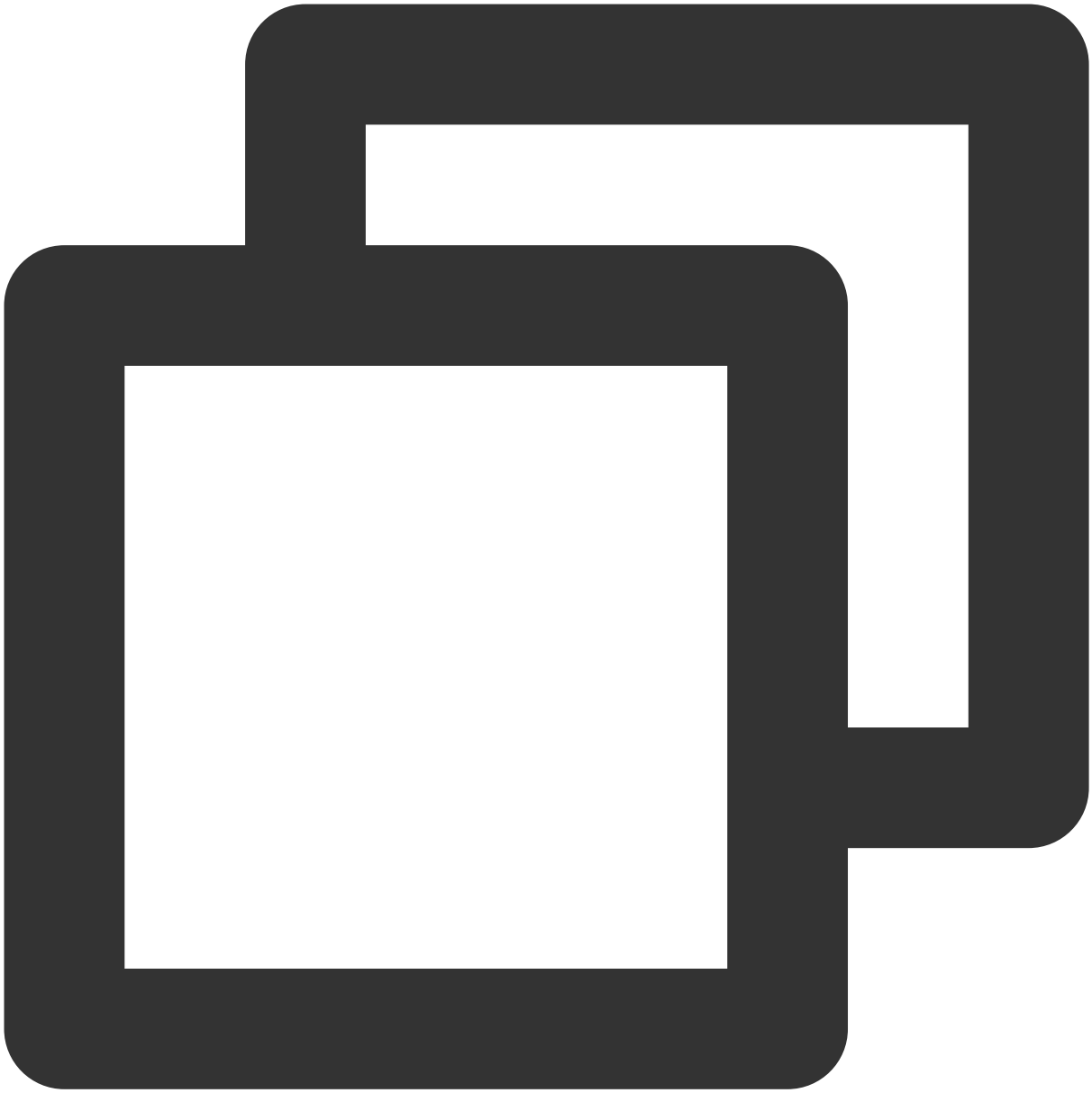


```
<template>
    <button @click="sdkCall">一键外呼</button>
</template>
<script>
export default {
   data() {
     phoneNumber: '19999999999' // 请替换为真实外呼号码
   },
   methods: {
     sdkCall() {
        window.tccc.Call.startOutboundCall({
```

```
            phoneNumber: this.phoneNumber,
          }).then((res) => {
            this.sessionId = res.data.sessionId;
          }).catch((err) => {
            const error = err.errorMsg;
          })
        }
    }
}
</script>
```



```
import { useState } from 'react';
export function CallButton() {
  const [phoneNumber, setPhoneNumber] = useState('19999999999') // 请替换为真实外呼号码

  function sdkCall(phoneNumber) {
    window.tccc.Call.startOutboundCall({
      phoneNumber,
    }).then((res) => {
      this.sessionId = res.data.sessionId;
    }).catch((err) => {
      const error = err.errorMsg;
    })
  }

  return (
    <button onClick={sdkCall}>一键外呼</button>
  )
}
```

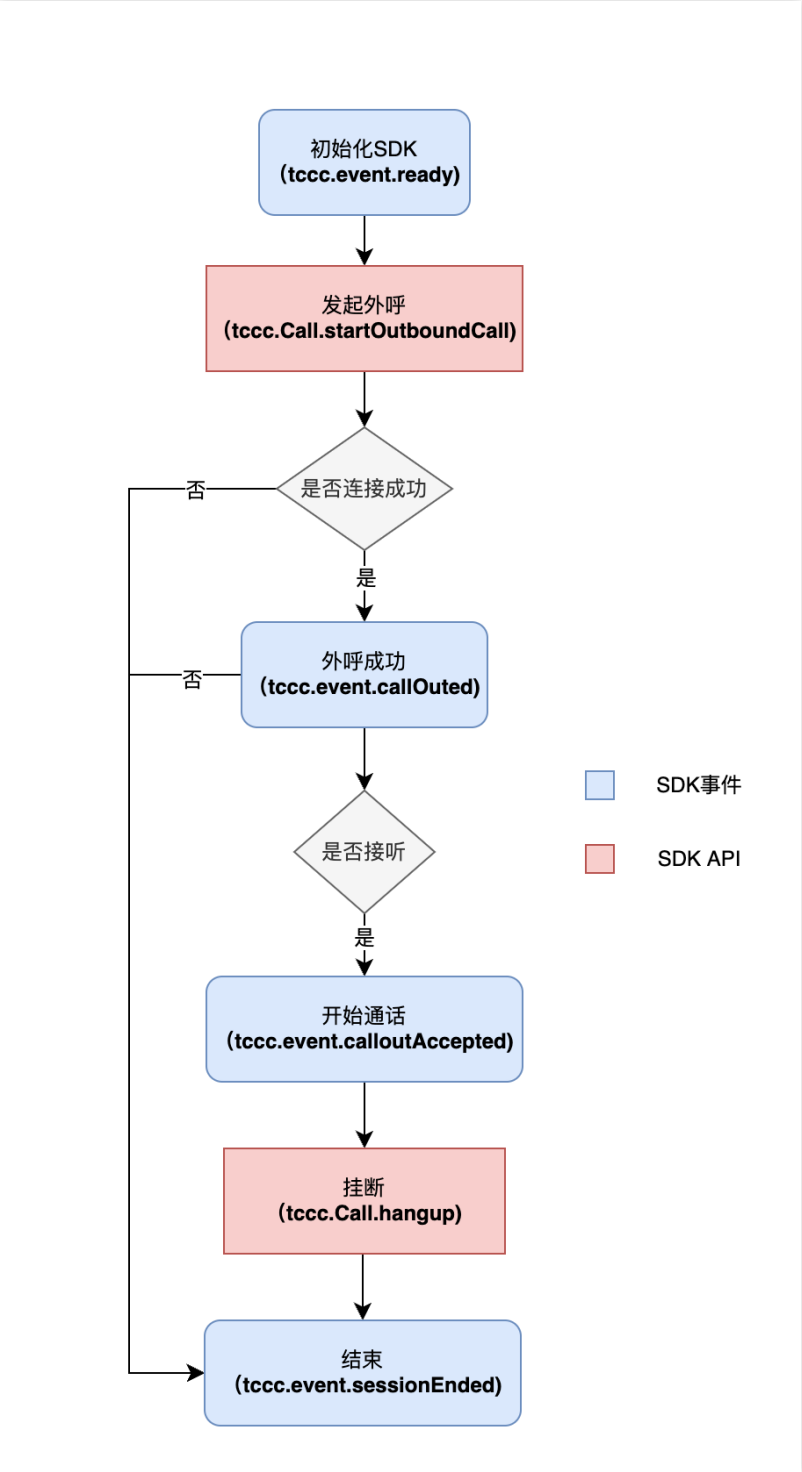


```
<button id="call">一键外呼</button>
<script>
    function sdkCall(phoneNumber) {
        window.tccc.Call.startOutboundCall({
            phoneNumber,            // 外呼号码
            phoneDesc: 'Tencent'    //备注文案，将会在通话条上替代号码的显示
        }).then((res) => {
            // 外呼成功，并获取外呼ID，后续可用作查询关联录音、服务记录信息
            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('19999999999');
      })
  </script>
```

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


初始化SDK
(tccc.event.ready)
发起外呼
(tccc.Call.startOutboundCall)
是否连接成功
否
是
外呼成功
(tccc.event.callOuted)
否
是否接听
是
开始通话
(tccc.event.calloutAccepted)
挂断
(tccc.Call.hangup)
结束
(tccc.event.sessionEnded)
SDK事件
SDK API

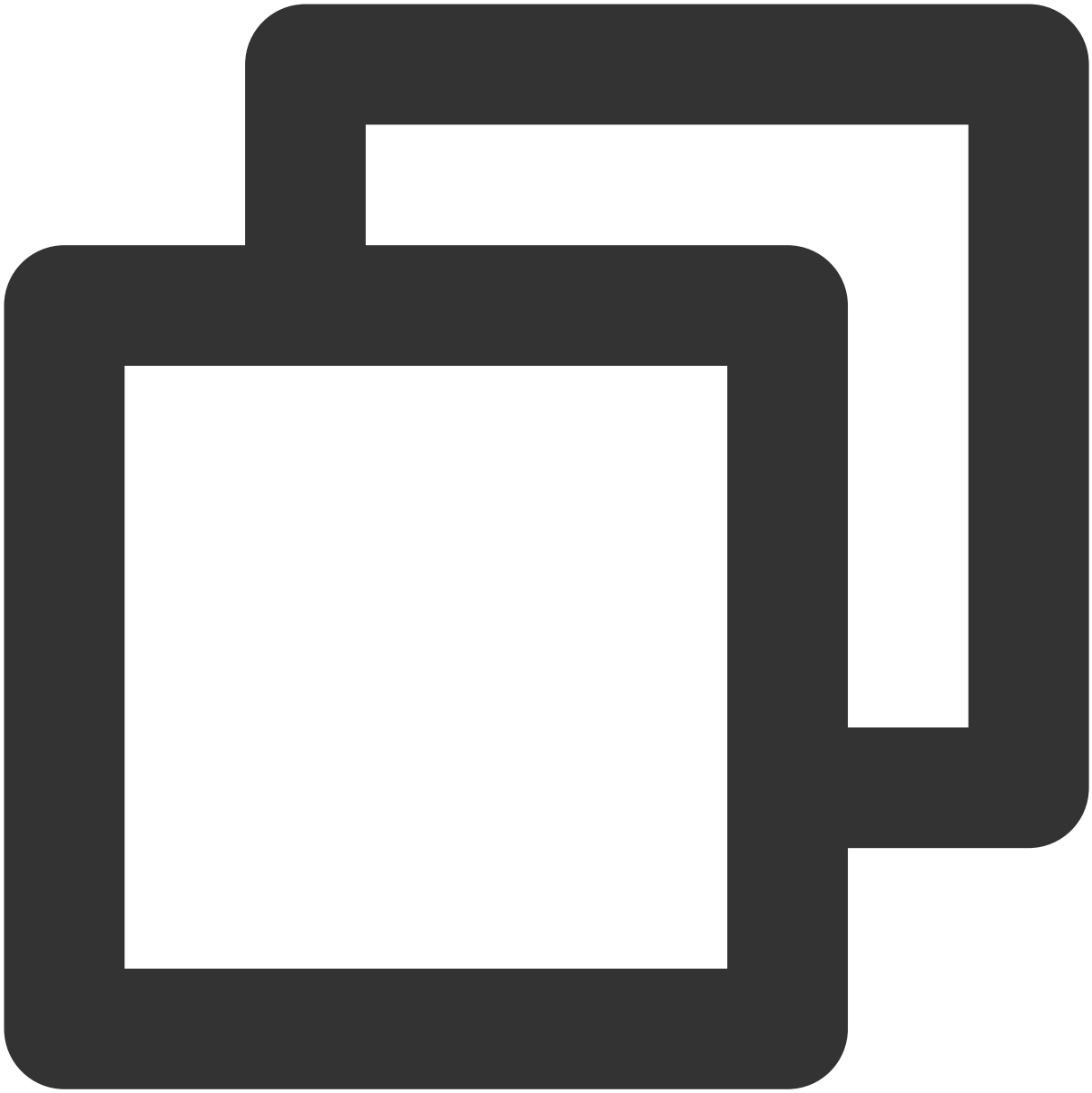
uni-app

最近更新时间：2024-04-01 18:09:16

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
  }
});
```

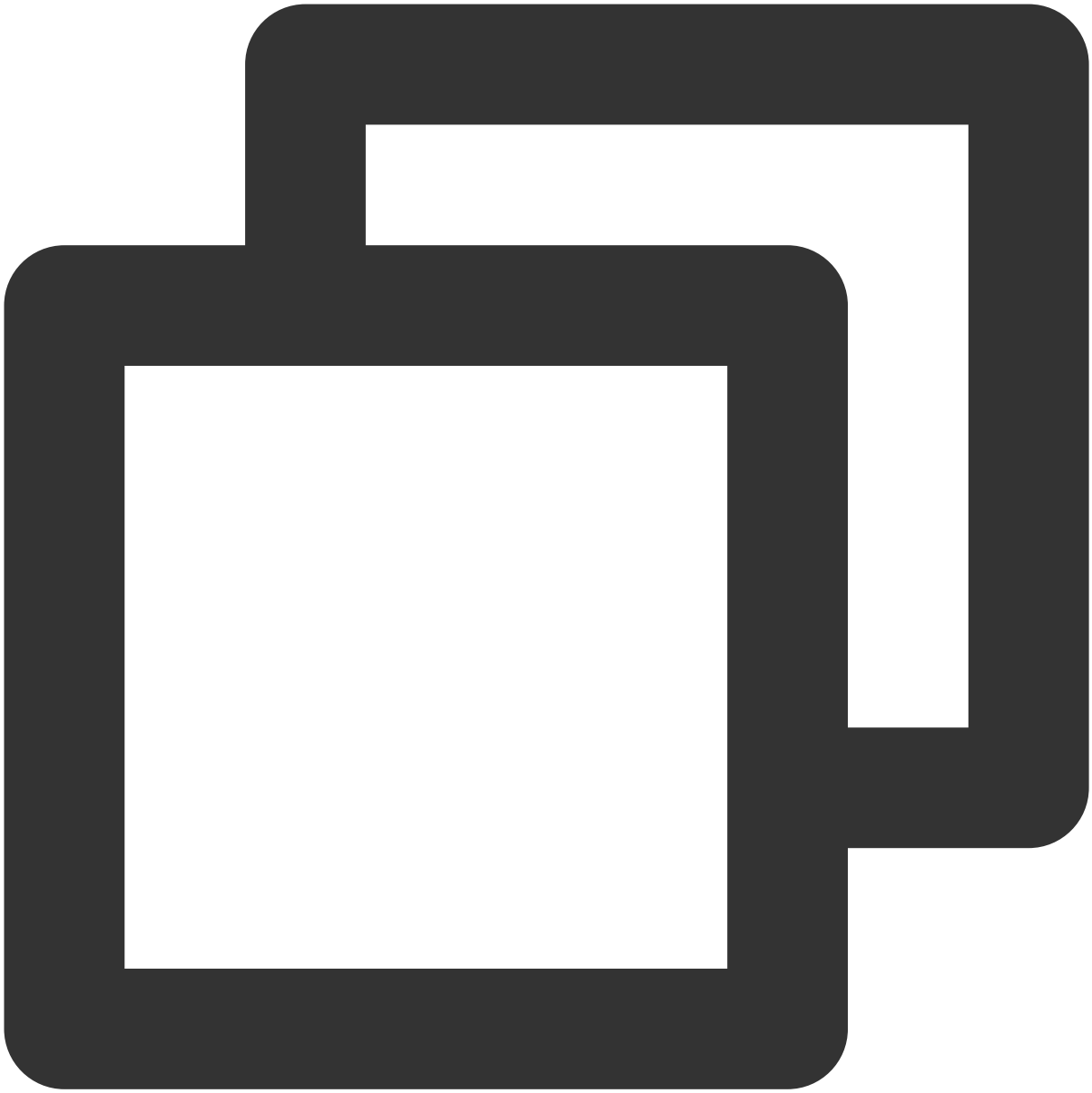
Android

最近更新时间：2024-04-01 18:09:56

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("");
```

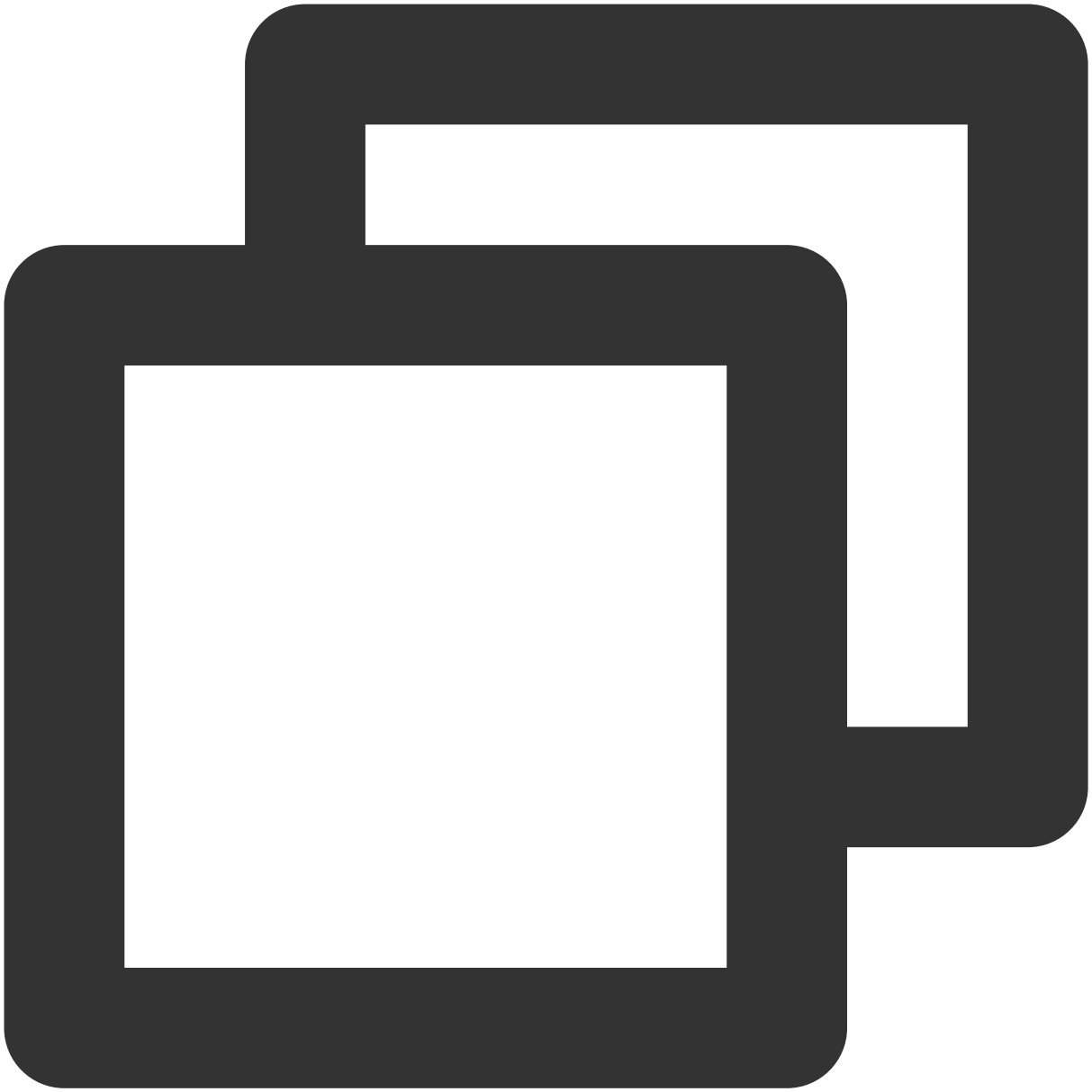
IOS

最近更新时间：2024-04-01 18:10:18

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:
    TCCCCommonCallback(NSString* funName) {
        mFunName = funName;
    }
    ~TCCCCommonCallback() 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;
//呼叫的手机号
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();
```

实现电话呼入

Web

最近更新时间：2024-04-01 18:13:20

初始化 SDK

请参见 初始化 SDK。

注意：

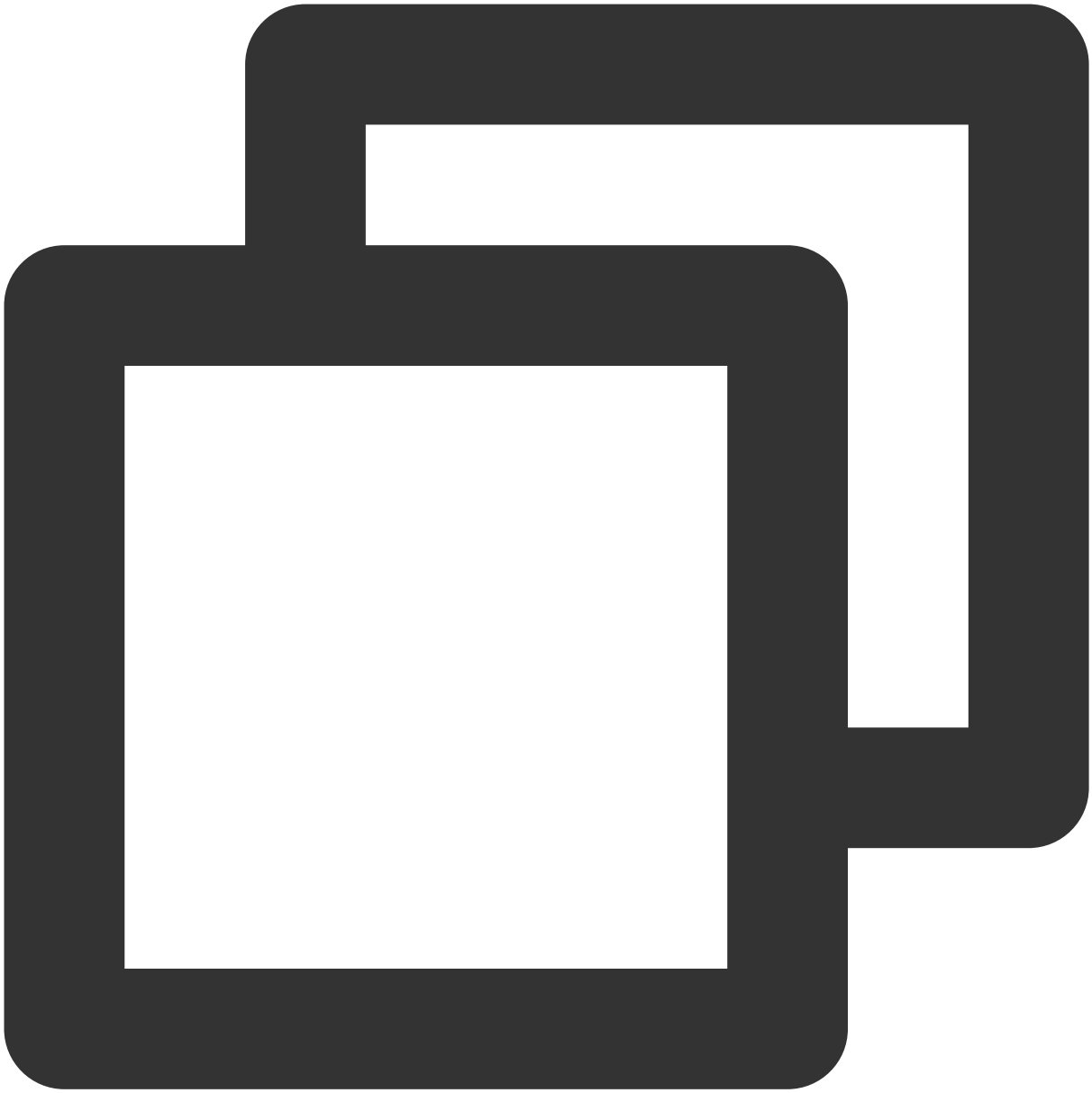
后续步骤需要在 tccc.events.ready 事件成功后才能执行。

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。

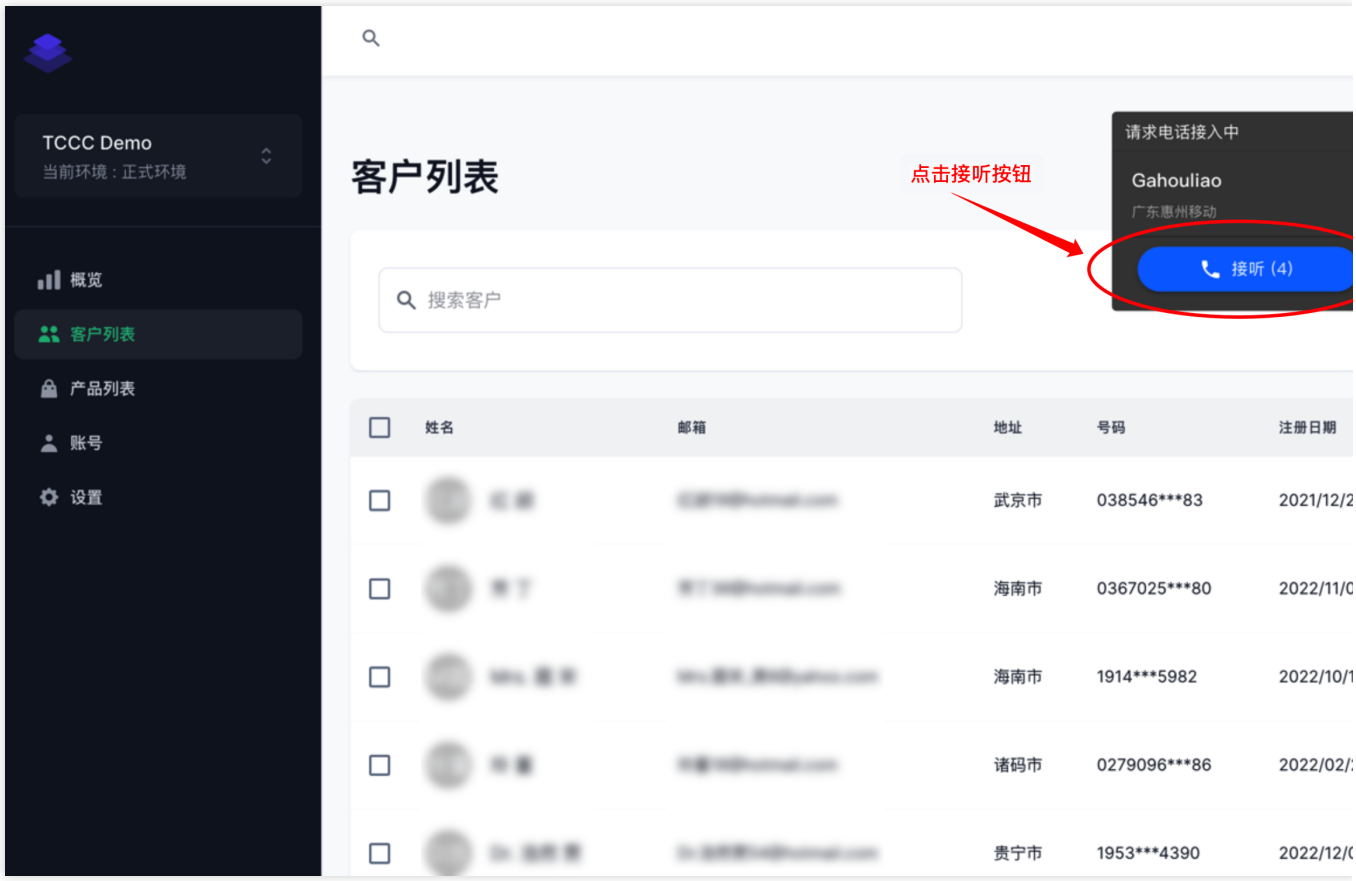
参考示例代码：



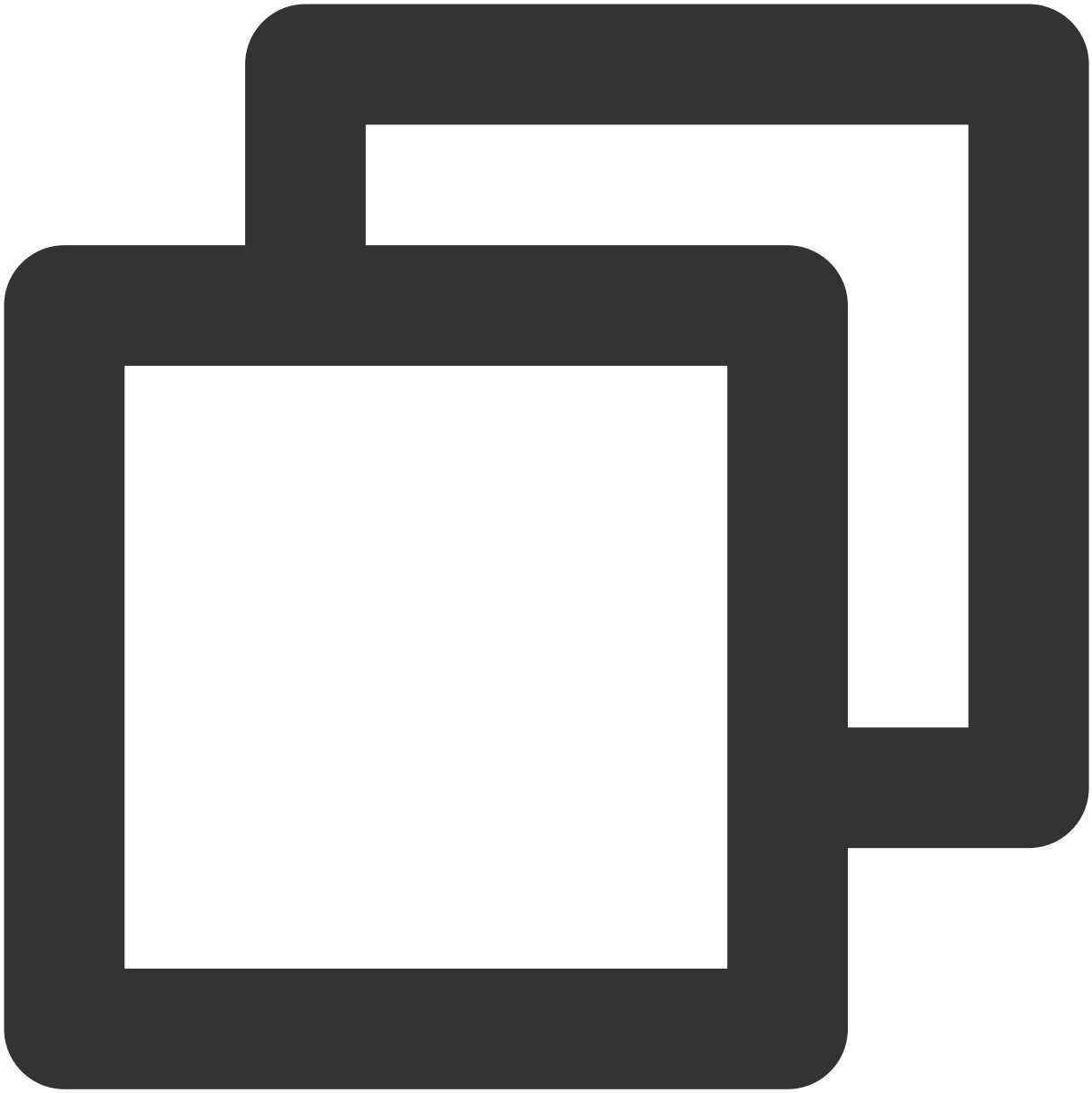
```
let sessionId; //存在公共区域，可以方便任意时候使用

// 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



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
})
```