多人音视频房间 SDK

更多特性

腾讯云

【联系我们】

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

文档目录

更多特性
设置昵称、头像（全平台）

最近更新时间：2024-08-02 16:24:31

本文介绍如何在 TUIRoomKit 中设置用户的头像和昵称。

设置头像、昵称

如果您需要自定义昵称或头像，可以使用如下接口进行更新：

Web&H5

```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl:
'http://xxx' });
```

Android

```
TUIRoomEngine.setSelfInfo("userName", "avatarUrl", null);
```

iOS

```
import TUIRoomEngine

TUIRoomEngine.setSelfInfo(userName: "xxx", avatarUrl: "xxx") {
    print("setSelfInfo success")
} onError: { code, message in
    print("setSelfInfo failed, code:\(code),message:\(message)")
}
```

uni-app

```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl:
'http://xxx' });
```

Flutter

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

TUIRoomEngine.setSelfInfo("userName", "avatarURL");
```

Electron

```
await TUIRoomEngine.setSelfInfo({ userName: 'jack', avatarUrl:
'http://xxx' });
```

注意：
由于用户隐私限制，昵称和头像更新可能会有延迟。如您需要更高的实时性，可以使用 会中修改昵称 功能。

会中修改昵称

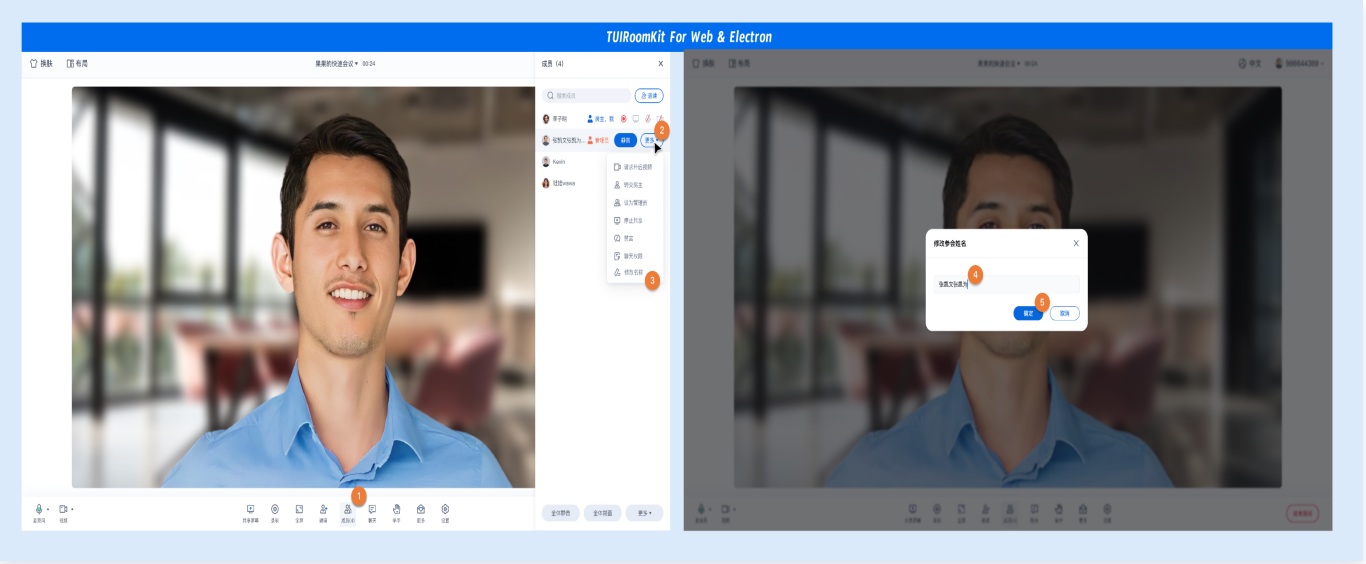
在会议中，参会人员可以实时修改自己的昵称，以方便在不同场景中展示不同的身份。修改后的昵称会即时生效，但仅限于当前会议。

说明：
会中改名特性需使用 TUIRoomKit v2.5.0 及以上版本，该特性目前**仅支持** Web、Electron、H5 和小程序端。

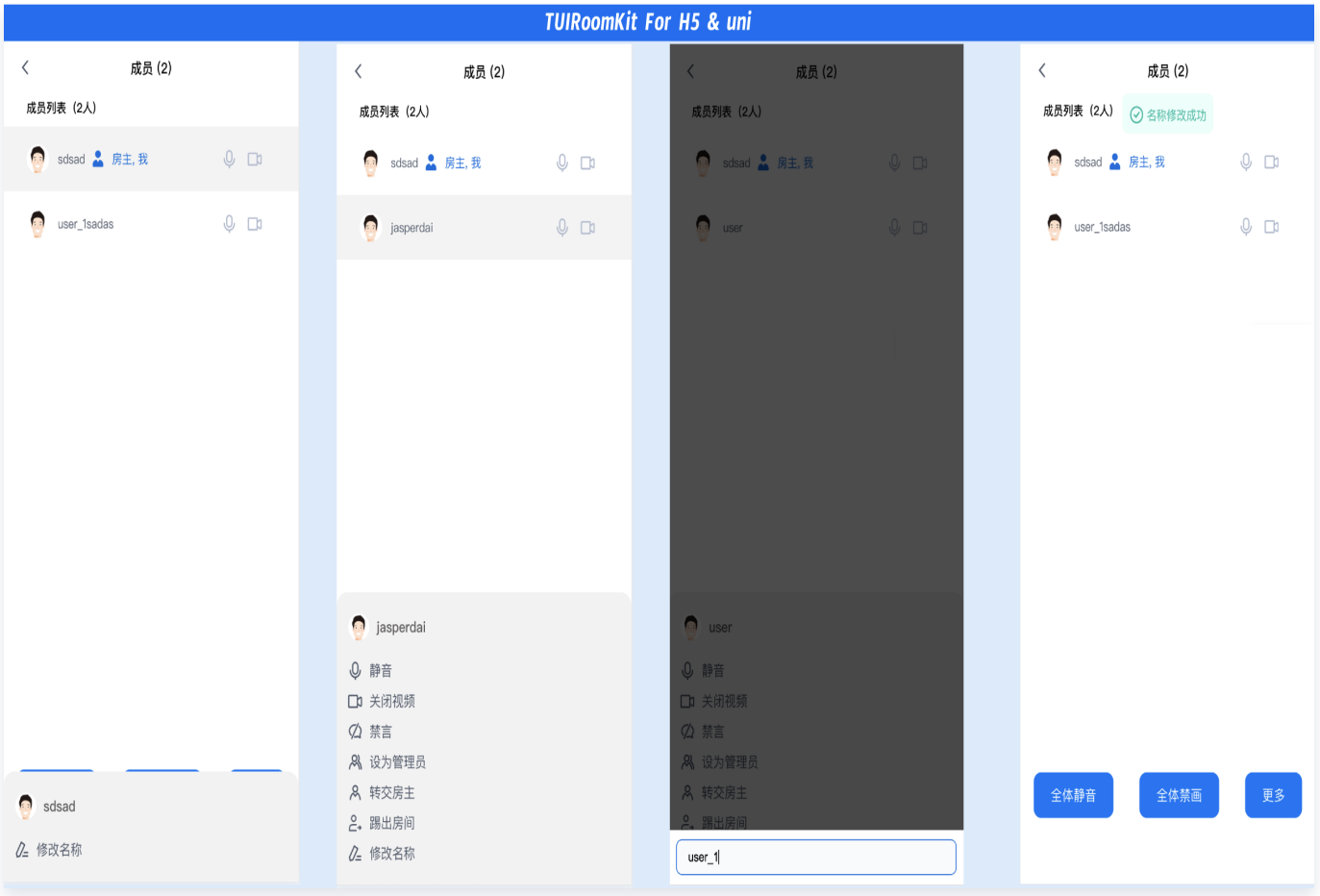
操作流程

1. 在 TUIRoomKit 内，会中点击底部工具栏**成员管理** > 选中自己或需要改名用户 > **更多** > **修改名称**；
2. 弹窗中输入需要修改后的名称后，点击确定即可即时生效。

Web&Electron



H5&小程序



操作权限

- 普通用户仅能修改自己的昵称。
- 房主或管理员可以修改自己或其他用户的昵称。

示例代码

如您需要在您的项目中，自行修改以支持会中修改昵称的功能，可使用如下 TUIRoomEngine 接口：

Web&H5

```
const roomEngine = TUIRoomEngine.getInstance();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
```

```
});
```

Electron

```
const roomEngine = TUIRoomEngine.getInstance();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
});
```

小程序

```
const roomEngine = new TUIRoomEngine();
await roomEngine.changeUserNameCard({
  userId: 'user_1234',
  nameCard: 'jack',
});
```

会中聊天
Web&Electron

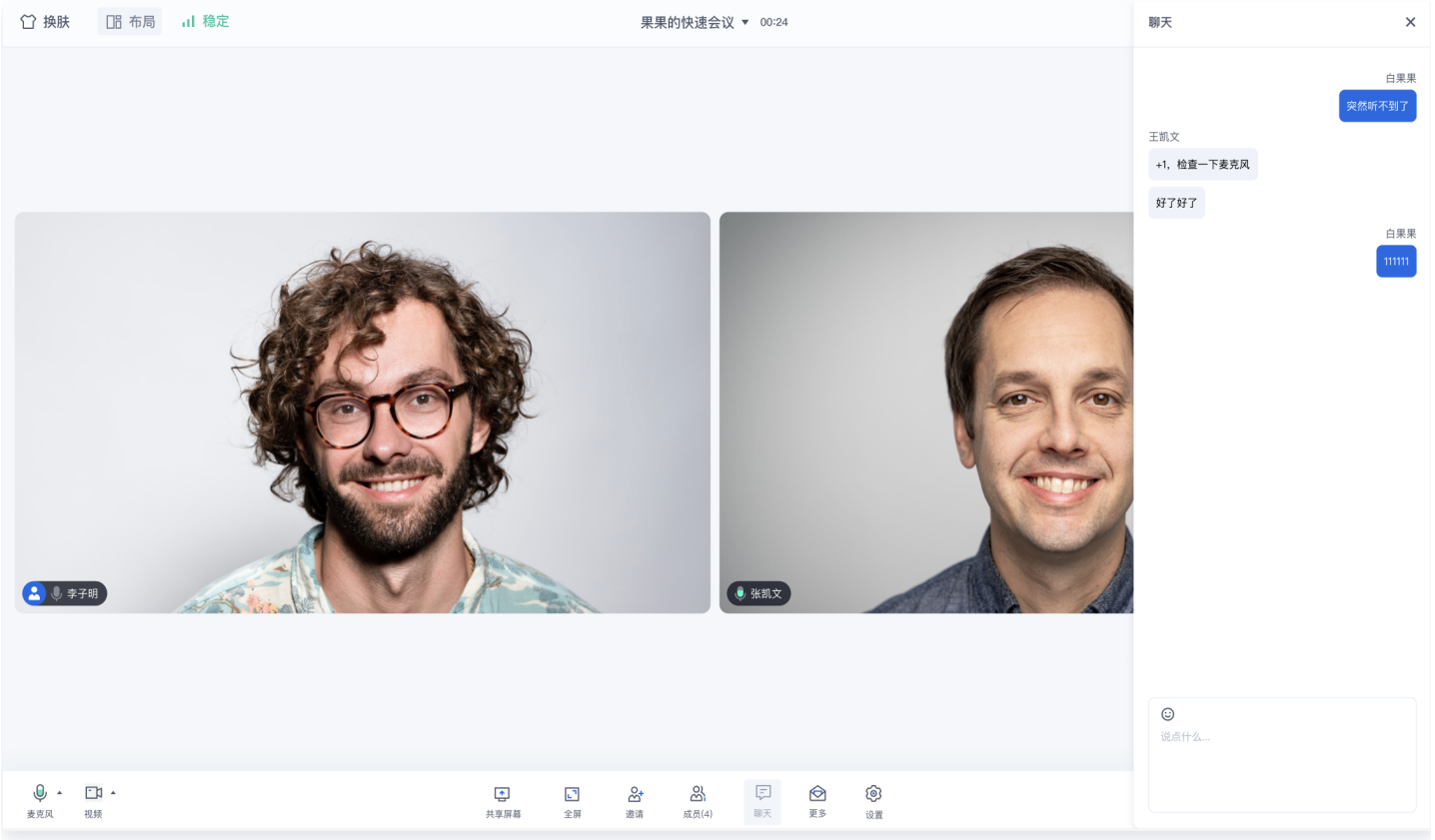
最近更新时间：2024-07-04 14:32:31

在视频会议中，参会者可以实时在聊天区发送消息，分享观点和想法，通过互发表情和动效氛围营造轻松愉快的交流环境。为了维护会议秩序，主持人或管理员可以设置禁止参会者在聊天中发送消息，以确保会议内容的专注和高效。通过灵活运用这些功能，视频会议可以为各种场景提供高效、便捷的沟通体验。

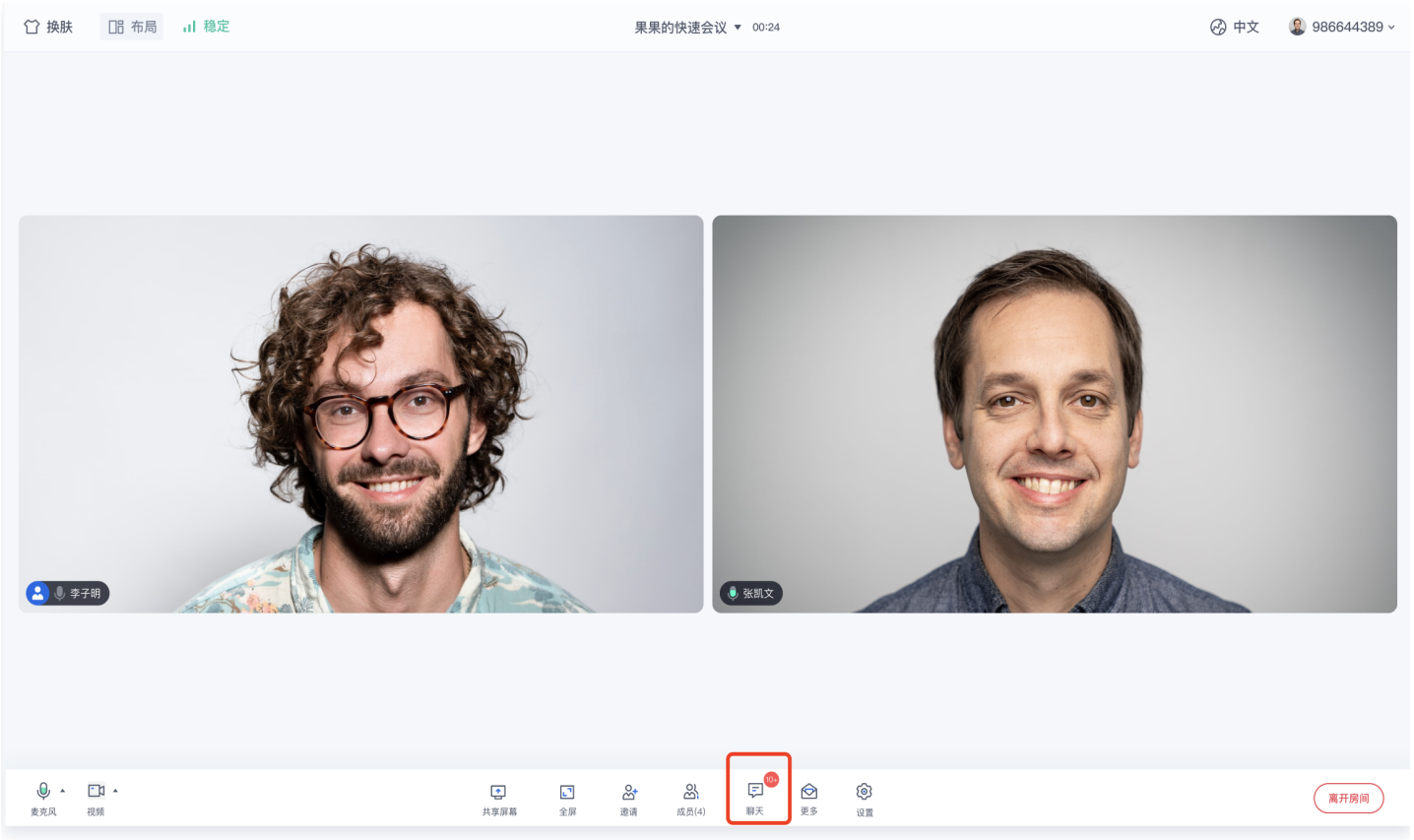
功能介绍

文字互动

在会议界面底部菜单栏中，您可以找到一个名为**聊天**的选项。点击它，会展开位于屏幕右侧的聊天框。在聊天框中，参会人员可以通过发送自定义文字信息与其他参会者进行沟通。这样的设计既方便了参会者之间的实时交流，又不会影响到正在发言的人。

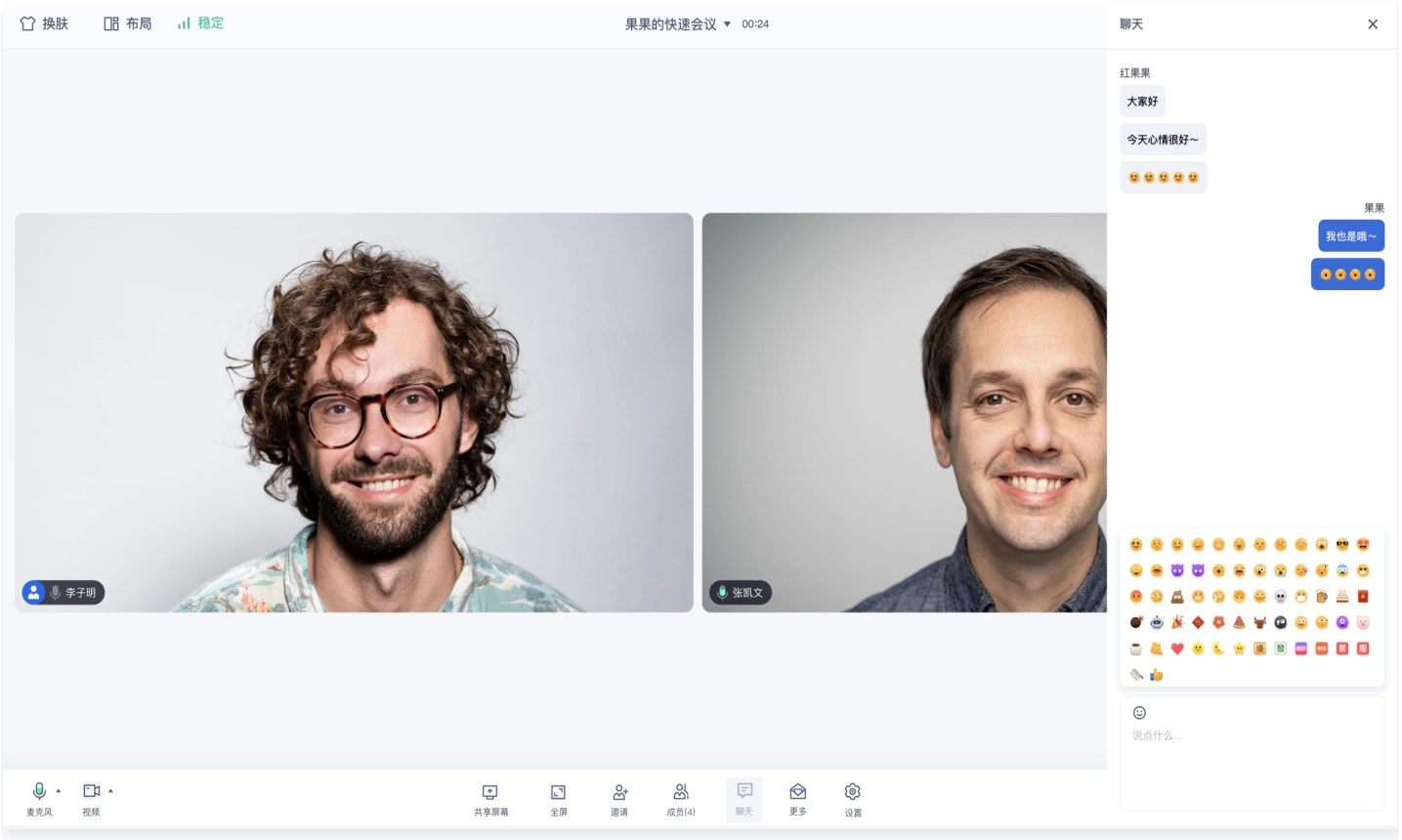


当有新的聊天信息发送时，底部工具栏 **聊天** 右上角会出现一个红色气泡提示，提醒您有未读信息。这样，您可以随时关注会议中的实时动态，确保不错过任何重要信息。



表情互动

点击聊天界面编辑消息栏中的**表情图标**，出现表情列表，点击对应表情即可进行发送，为演讲人的精彩发言点赞。

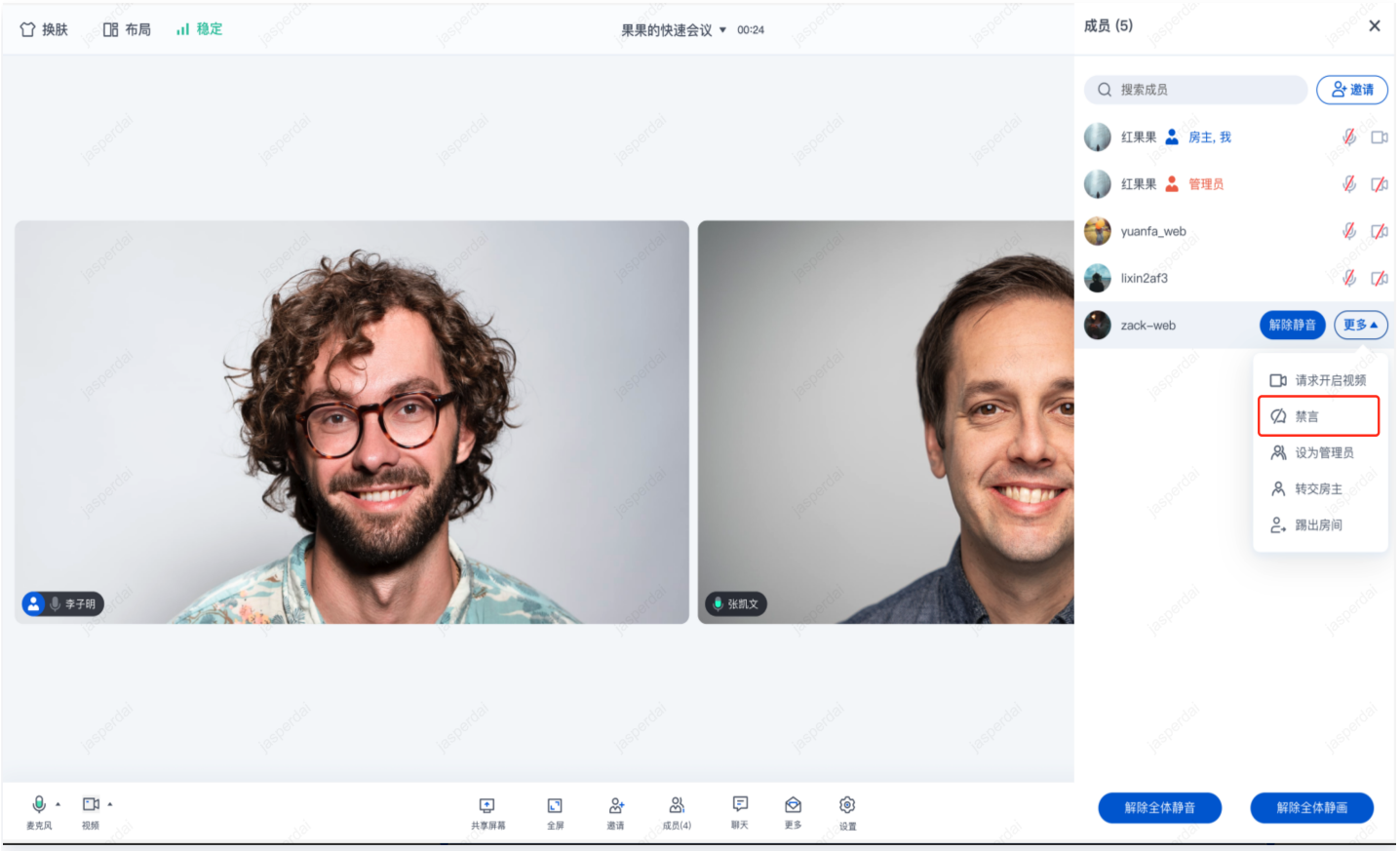


说明：

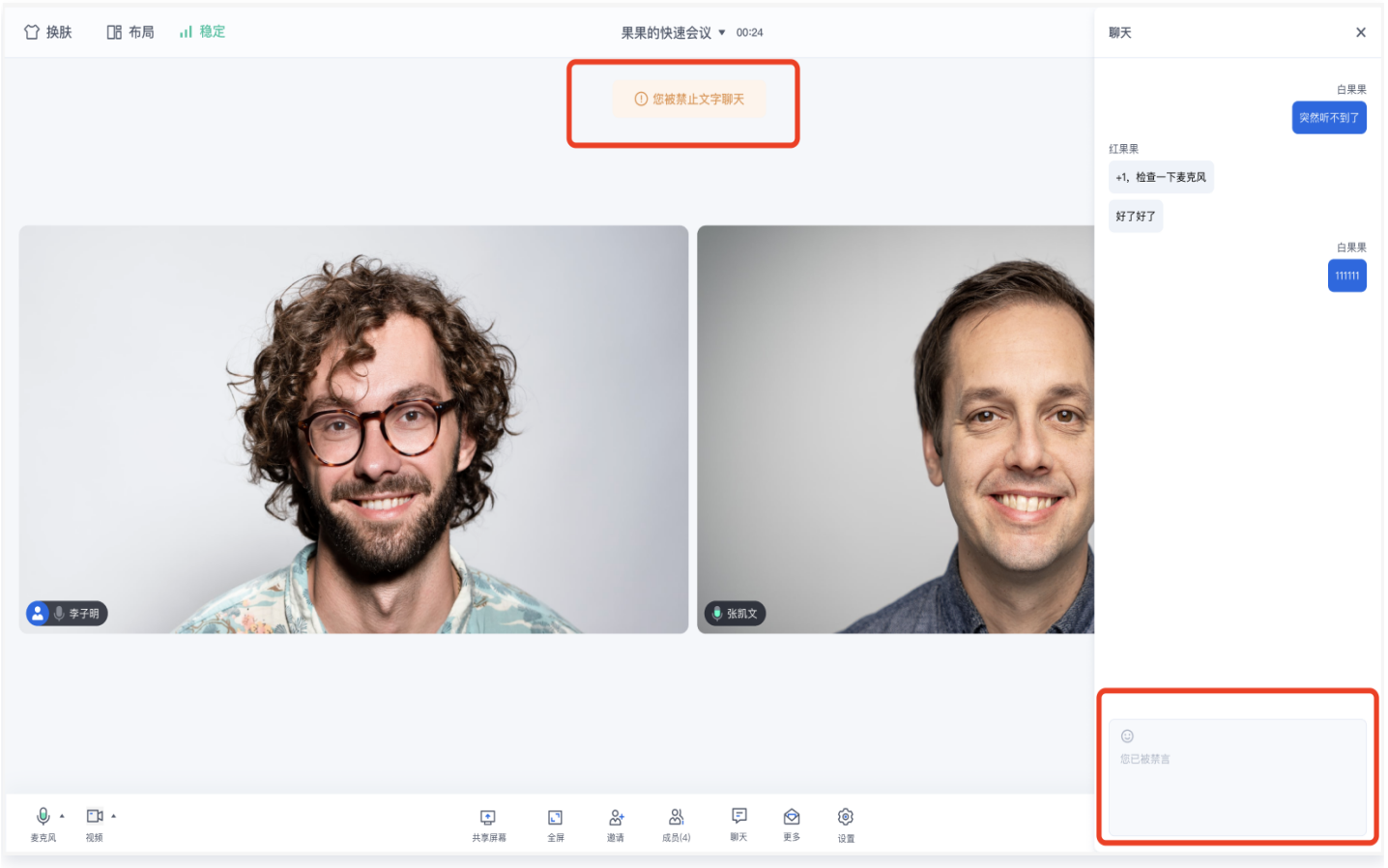
为尊重表情设计版权，TUIRoomKit 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。默认的**小黄脸表情包版权归腾讯云所有**，可有偿授权使用，如您希望获得授权可 提交工单 联系我们。

管控会中聊天权限

主持人/管理员可在成员管理中设置某一成员的聊天权限。



普通成员受到禁止发言，无法进行文字或表情的编辑和发送。



功能接入

目前 Web 和 Electron 的 TUIRoomKit 内部已经集成会中的聊天能力，您可以单击 Github 下载 TUIRoomKit 代码，并参见代码仓库 README.md 文档跑通 TUIRoomKit Web 示例工程。

> **注意：**
> TUIRoomKit Web&Electron 目前仅支持文本信息聊天，如需发送图片信息聊天等扩展内容，可通过获取 tim 实例的方式进行实现，web 端 tim 能力请查看： IM API 文档 。

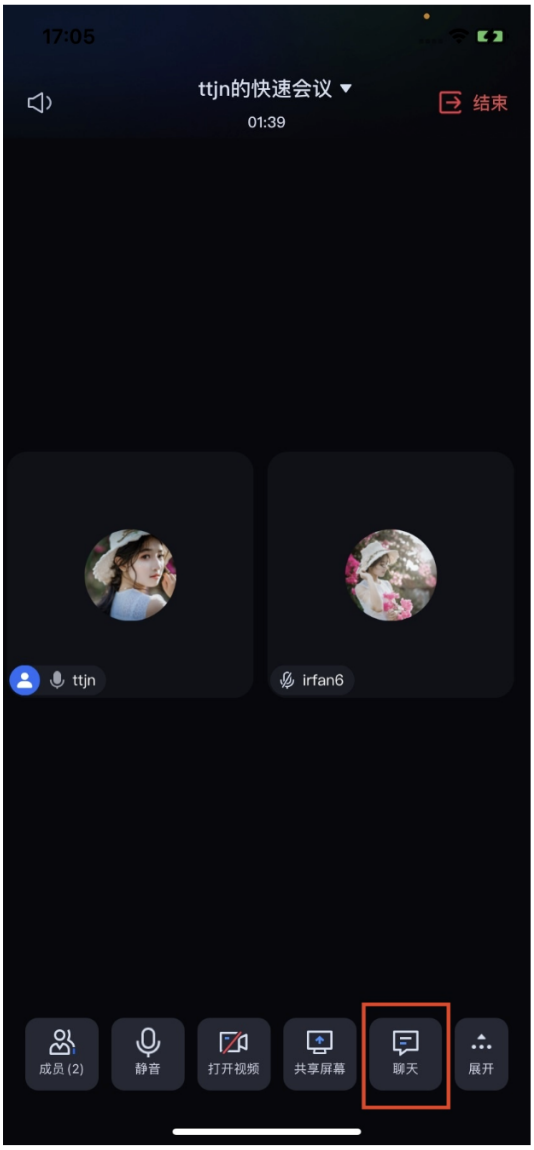
Android&iOS

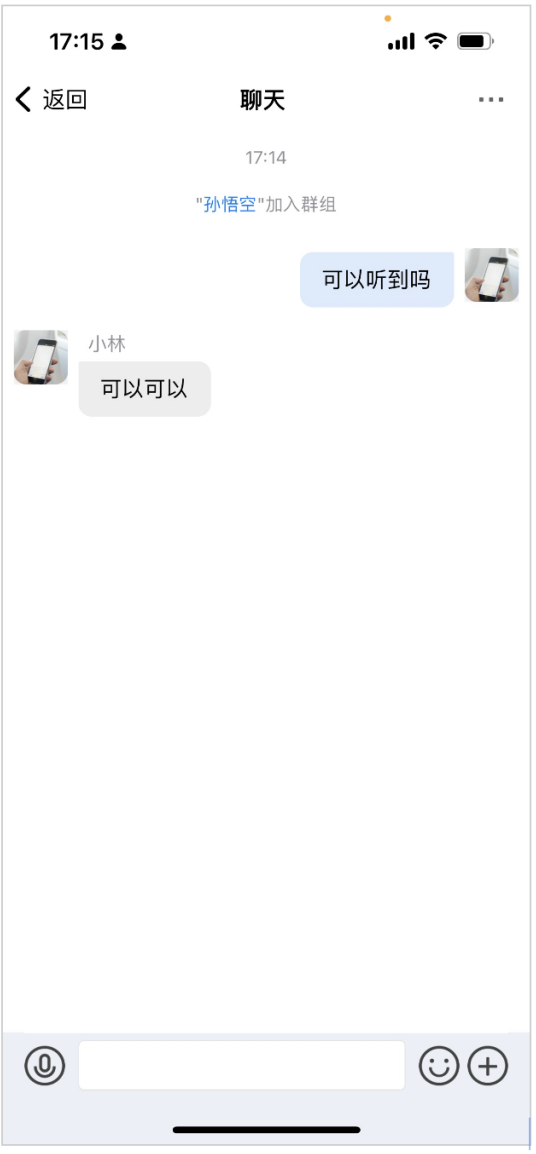
最近更新时间：2024-07-04 14:32:31

在视频会议中，参会者可以实时在聊天区发送消息，分享观点和想法，通过互发表情和动效氛围营造轻松愉快的交流环境。为了维护会议秩序，主持人或管理员可以设置禁止参会者在聊天中发送消息，以确保会议内容的专注和高效。通过灵活运用这些功能，视频会议可以为各种场景提供高效、便捷的沟通体验。

文字互动

在会议界面底部菜单栏中，您可以找到一个名为**聊天**的选项。点击它，会跳转到聊天页面。在聊天页面，参会人员可以通过发送自定义文字信息与其他参会者进行沟通。这样的设计既方便了参会者之间的实时交流，又不会影响到正在发言的人。





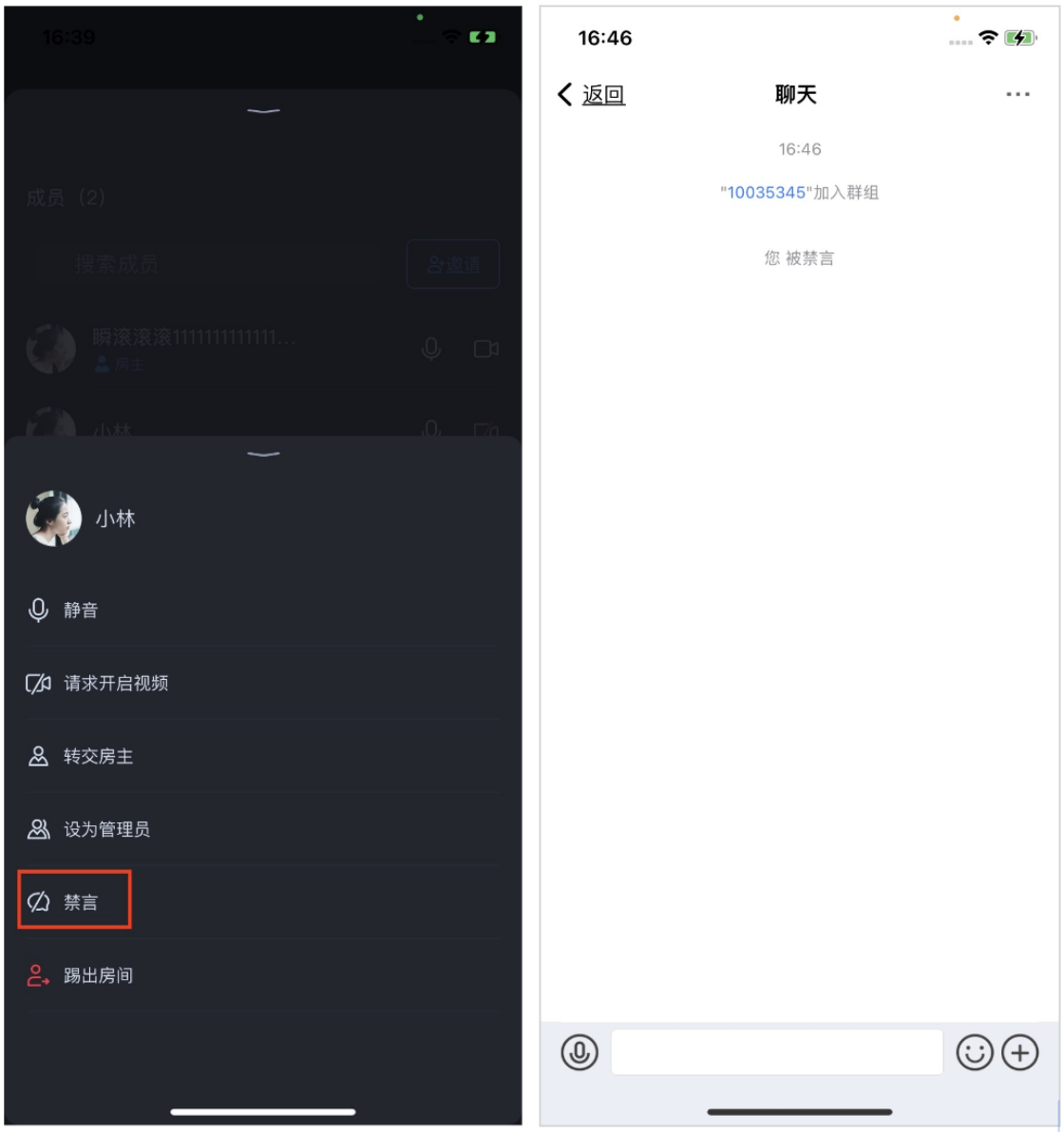
表情互动

点击聊天界面编辑消息栏中的 **表情图标**，出现表情列表，点击对应表情即可进行发送，为演讲人的精彩发言点赞。

> **说明：**
> 为尊重表情设计版权，TUIRoomKit 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。默认的**小黄脸表情包版权归腾讯云所有**，可有偿授权使用，如您希望获得授权可 提交工单 联系我们。

管控会中聊天权限

主持人/管理员可在成员管理中设置某一成员的聊天权限。如果被禁言，普通成员无法进行文字或表情的编辑和发送。



接入管理

Android

module 源码集成

1. 在 Github 中克隆/下载代码，然后拷贝Android目录下的tuichat子目录到您当前工程中的 app 同级目录中。
2. 工程根目录下找到setting.gradle文件，并在其中增加如下代码，它的作用是将tuichat作为本地模块导入到您当前的项目中。

```
include ':tuichat'
```

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

```
api project(':tuichat')
```

4. 添加 maven 仓库 和 Kotlin 支持，在 root 工程的 build.gradle 文件（与 settings.gradle 同级）中添加：

```
buildscript {
    repositories {
        mavenCentral()
        maven { url
"https://mirrors.tencent.com/nexus/repository/maven-public/" }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.0.0'
        classpath "org.jetbrains.kotlin:kotlin-gradle-
plugin:1.5.31"
    }
}
```

如果您使用 Gradle 8.x，则需要添加以下代码。

```
buildscript {
    repositories {
        mavenCentral()
        maven { url
"https://mirrors.tencent.com/nexus/repository/maven-public/" }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:8.0.2'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.9.0"
    }
}
```

说明：

> Kotlin 、Gradle 和 AGP 的版本对应关系可以 在此查看 。

iOS

接入聊天挂件

使用 CocoaPods 导入聊天挂件，具体步骤如下：

1. 在您的 `Podfile` 文件中添加以下依赖。

```
pod 'TUIChat'          # [可选] 聊天挂件
```

2.执行以下命令，安装组件。

```
pod install
```

Flutter

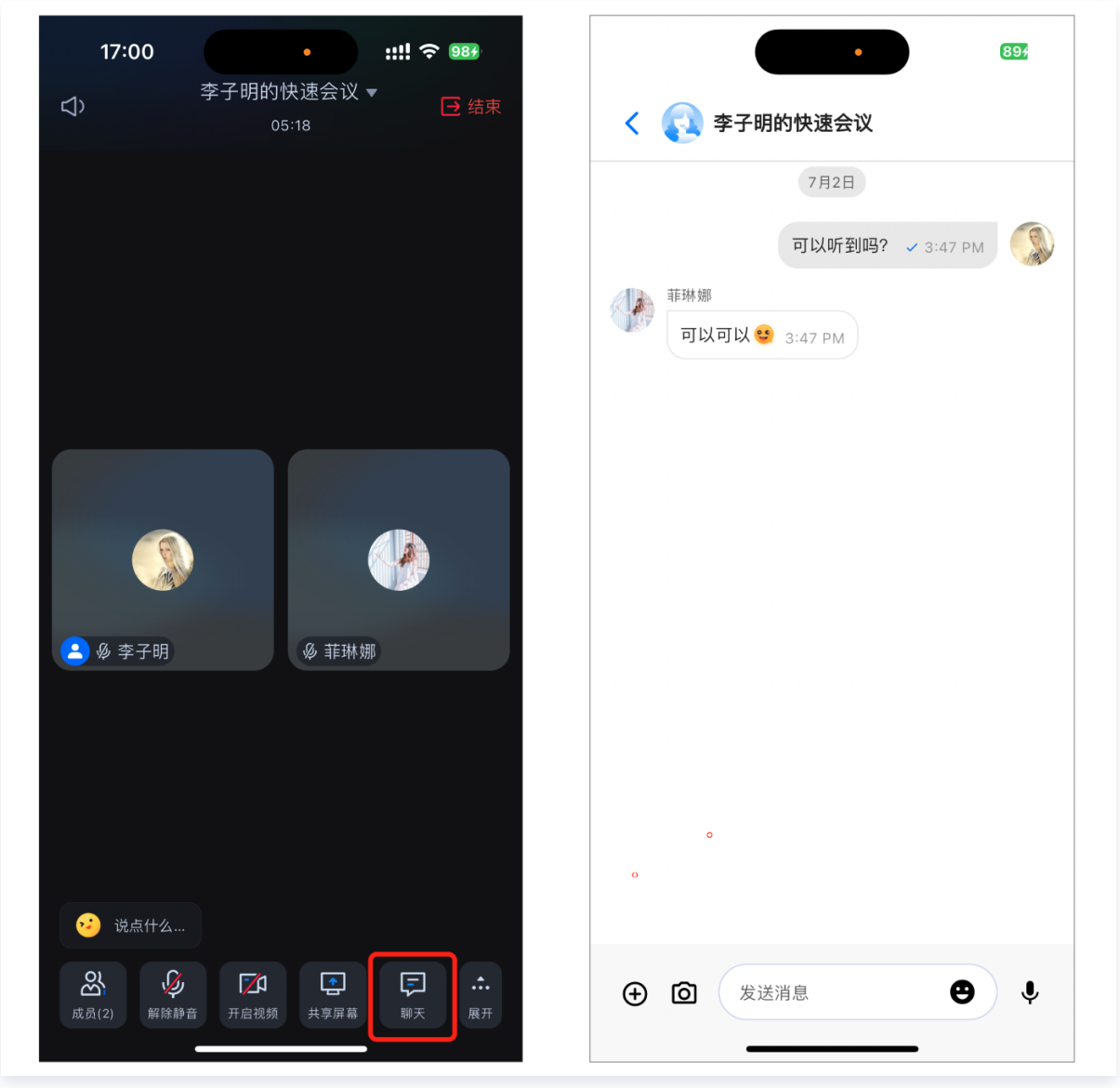
最近更新时间：2024-07-29 11:17:11

在视频会议中，参会者可以实时在聊天页面发送消息，分享观点和想法，通过互发表情和动效氛围营造轻松愉快的交流环境。为了维护会议秩序，主持人或管理员可以设置禁止参会者在聊天中发送消息，以确保会议内容的专注和高效。通过灵活运用这些功能，视频会议可以为各种场景提供高效、便捷的沟通体验。

功能介绍

文字、多媒体信息互动

单击会议界面底部**聊天**选项即可到达聊天界面。参会者可自由发送**文字**、**图片**、**视频**、**语音**，实时沟通无阻，且不影响会议发言进程。



表情互动

单击聊天界面编辑消息栏中的**表情图标**，出现表情列表，单击对应表情即可出现在消息栏中进行发送。

管控会中聊天权限

主持人/管理员可在**成员管理**中设置某一成员的聊天权限。如果被禁言，普通成员将无法进行消息的发送。

功能接入

集成聊天组件

在您的工程 `pubspec.yaml` 文件中，添加 tencent_cloud_chat_message 插件依赖。

```
dependencies:
    tencent_cloud_chat_message: 最新版本
```

国际化语言配置

此步骤为必需配置。首先，将本地化工具导入到应用程序的入口文件中。

```
import
'package:tencent_cloud_chat_intl/localizations/tencent_cloud_chat_locali
```

```
zations.dart';
```

接下来，将本地化配置添加到 `MaterialApp` 或 `GetMaterialApp` 等第三方包提供的其他条目中，这里以 `GetMaterialApp` 为例：

```
GetMaterialApp(
  localizationsDelegates: const [
    /// 您原先的配置
    GlobalMaterialLocalizations.delegate,

    /// 添加此行
    ...TencentCloudChatLocalizations.localizationsDelegates,
  ],
  supportedLocales: [
    /// 您原先的配置
    ...S.delegate.supportedLocales,

    /// 添加此行
    ...TencentCloudChatLocalizations.supportedLocales,
  ],
  /// 其他配置
)
```

初始化和登录

在您的项目中添加如下代码，它的作用是通过调用聊天中的相关接口完成组件的初始化及登录。这个步骤非常关键，因为只有在初始化后才能正常使用聊天，故请您耐心检查相关参数是否配置正确。其中的参数 **sdkAppId**、**userID** 和 **userSig** 您已在 登录TUIRoomKit 时使用过。

```
import
'package:tencent_cloud_chat/components/component_config/tencent_cloud_ch
at_message_common_defines.dart';
import
'package:tencent_cloud_chat/components/component_config/tencent_cloud_ch
at_message_config.dart';
import
'package:tencent_cloud_chat/models/tencent_cloud_chat_models.dart';
import 'package:tencent_cloud_chat/tencent_cloud_chat.dart';
import
'package:tencent_cloud_chat_message/tencent_cloud_chat_message.dart';

await TencentCloudChat.controller.initUIKit(
```

```
    options: TencentCloudChatInitOptions(
      sdkAppID: 'SDKAPPID',   // 您的SDKAPPID
      userID: 'userID',       // 您的userID
      userSig: 'userSig',     // 您的UserSig
    ),
    components: TencentCloudChatInitComponentsRelated(
      usedComponentsRegister: [TencentCloudChatMessageManager.register],
// 注册聊天组件
      componentConfigs: TencentCloudChatComponentConfigs(
        messageConfig: TencentCloudChatMessageConfig(
          // 以下代码中的配置为建议配置。
          showMessageSenderName: ({groupID, topicID, userID}) => true,
          showSelfAvatar: ({groupID, topicID, userID}) => true,
          defaultMessageMenuConfig: ({groupID, topicID, userID}) =>
              TencentCloudChatMessageDefaultMessageMenuConfig(
            enableMessageForward: false,
            enableMessageSelect: false,
          ),
        ),
      ),
    ),
    plugins: [],
);
```

使用表情（可选）

如您需要使用发送和接收表情，您需要进行如下配置：

添加依赖

在您工程的 `pubspec.yaml` 文件中，添加 tencent_cloud_chat_sticker 插件依赖。

完成配置

在上一步 初始化和登录 的 `plugins` 中，添加如下代码：

```
plugins: [
  TencentCloudChatPluginItem(
    name: "sticker",
    initData: TencentCloudChatStickerInitData(
      useDefaultSticker: true,             // 默认表情，仅此表情包可与
TUIRoomKit其他端互通。
      useDefaultCustomFace_4350: false,    // 如您无需使用其他端TUIRoomKit，
可启用下列表情包。更多详情请查看下方说明。
```

```
      useDefaultCustomFace_4351: false,
      useDefaultCustomFace_4352: false,
      customStickerLists: [],              // 如您需要添加自定义的表情包，请在
此添加。
      userID: 'userId',                    // 您的userId
    ).toJson(),
    pluginInstance: TencentCloudChatStickerPlugin(
      context: context,
    ),
  ),
],
```

说明：
为尊重表情设计版权，TUIRoomKit example 工程中不包含大表情元素切图，正式上线商用前请您替换为自己设计或拥有版权的其他表情包。默认的**小黄脸表情包版权归腾讯云所有**，可有偿授权使用，如您希望获得授权可 提交工单 联系我们。

使用聊天

当您 创建会议或加入会议成功 时，您需要向会议界面 `ConferenceMainPage` 中传入 `chatWidget` 。传入后将显示底部工具栏中的聊天按钮，点击聊天按钮后自动导航到聊天页面。

```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => ConferenceMainPage( // 会议主界面
      chatWidget: TencentCloudChatMessage(
          options: TencentCloudChatMessageOptions(groupID:
'yourConferenceId'),  //您的ConferenceId
      ),
    ),
  ),
);
```

完成以上配置后，您便可以在会议中点击聊天按钮进行聊天交流。

离线唤醒
会前提醒（Android/iOS）

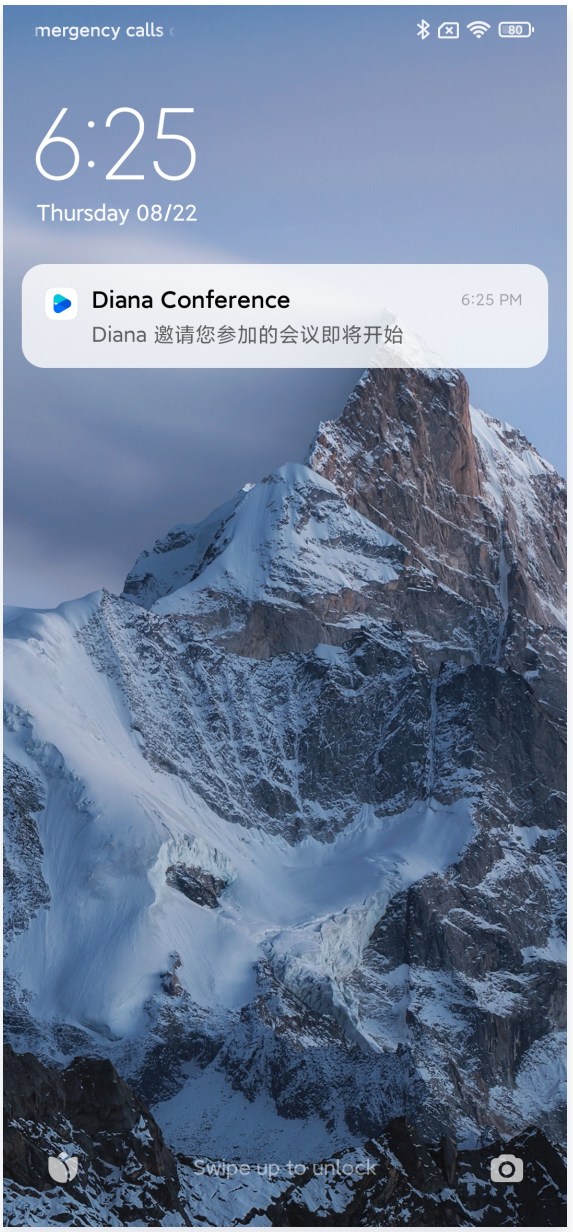
最近更新时间：2024-08-29 17:20:11

当您成功预定会议后，TUIRoomKit 组件将在会议开始前10分钟向参会成员推送会前提醒消息。本文档将指引您接入 TIMPush 推送插件，以开启会前提醒功能。

接入效果

TUIRoomKit 接入 TIMPush 推送插件的效果如下（以小米手机 Redmi Note 8 Pro 的显示效果为例）：

应用在后台时或离线时	锁屏时



准备条件

在进行离线推送功能的集成前，请确保已按照官方文档完成 厂商配置，以确保 TUIRoomKit 的离线推送功能正常运行。

功能接入

Android

1. 请参见推送插件 TIMPush 快速接入文档，完成除步骤6以外的所有步骤（TUIRoomKit 组件内部已经进行会前提醒的离线消息推送，所以步骤6不需要单独配置）。

2. (可选) 若您想实现点击通知立即进房，可参考以下代码，注册回调时机建议放在应用 Application 的 onCreate() 函数中：

```
TUICore.registerEvent(TUIConstants.TIMPush.EVENT_NOTIFY,
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION, new
ITUINotification() {
    @Override
    public void onNotifyEvent(String key, String subKey,
Map<String, Object> param) {
        if (TUIConstants.TIMPush.EVENT_NOTIFY.equals(key) &&
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION.equals(subKey)) {
            if (param != null) {
                String extString = (String)
param.get(TUIConstants.TIMPush.NOTIFICATION_EXT_KEY);
                try {
                    // 可根据您的业务，点击通知跳转到相应的界面，以下是跳
转到房间的示例。
                    JSONObject roomObject = new
JSONObject(extString);
                    String roomId =
roomObject.getString("RoomId");
                    if (!TextUtils.isEmpty(roomId)) {
                        loginAndEnterRoom(roomId);
                    }
                } catch (Exception e) {
                }
            }
        }
    }
});

private void loginAndEnterRoom(String roomId) {
   int sdkAppId = "您的appId";
   String userId = "您的UserId"
   String userSig = GenerateTestUserSig.genTestUserSig(userId);

   // 登录
   TUILogin.login(this.getApplicationContext(), sdkAppId, userId,
userSig, new TUICallback() {
       @Override
       public void onSuccess() {
           // 登录成功后进入房间
```

```
            ConferenceDefine.JoinConferenceParams params = new
ConferenceDefine.JoinConferenceParams(roomId);
            Intent intent = new Intent(getApplicationContext(),
ConferenceMainActivity.class);
            intent.putExtra(KEY_JOIN_CONFERENCE_PARAMS, params);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            this.startActivity(intent);
        }

        @Override
        public void onError(int errorCode, String errorMessage) {
        }
    });
  }
```

> **注意：**
> 若您想实现点击推送进入房间，**必须进房前完成登录**。

iOS

1. 集成 TIMPush 组件

```
pod 'TIMPush', '8.1.6108'
```

2. 配置推送参数

完成 厂商配置 后，可以在 即时通信 IM 控制台 得到证书 ID。您需要在 AppDelegate 中，实现 `offlinePushCertificateID` 协议方法并返回证书 ID 。

Swift

```
import TIMPush

extension AppDelegate: TIMPushDelegate {
    func offlinePushCertificateID() -> Int32 {
        return kAPNSBusiId
    }
}
```

OC

```
#import "TIMPush/TIMPushManager.h"

@interface AppDelegate () <TIMPushDelegate>

- (int)offlinePushCertificateID {
    return kAPNSBusiId;
}
```

3. 点击离线推送后自定义跳转（可选）

在默认情况下，点击通知会跳转到 app。您可以参考以下代码实现点击通知立即进入会议，也可以查看 github 中的 SceneDelegate 和 AppDelegate 文件。
如果是冷启动，需要在 SceneDelegate 中解析通知消息，得到要进入会议的 roomId。

Swift

```
import UIKit

class SceneDelegate: UIResponder, UIWindowSceneDelegate {
    var window: UIWindow?

    func scene(_ scene: UIScene,
               willConnectTo session: UISceneSession,
               options connectionOptions: UIScene.ConnectionOptions)
{
        guard let windowScene = (scene as? UIWindowScene) else {
return }
        window = UIWindow(windowScene: windowScene)
        let loginVC = TRTCLoginViewController() //您自己的登录页面
        loginVC.roomId = processOfflinePush(connectionOptions:
connectionOptions)
        let nav = UINavigationController(rootViewController:
loginVC)
        window?.rootViewController = nav
        window?.makeKeyAndVisible()
    }

    private func processOfflinePush(connectionOptions:
UIScene.ConnectionOptions) -> String? {
```

```
        guard let pushNotification =
connectionOptions.notificationResponse?.notification.request.content
.userInfo else { return nil }
        guard let extString = pushNotification["ext"] as? String
else { return nil }
        guard let dict = extString.convertToDic() else { return nil
}
        return dict["RoomId"] as? String
    }
```

OC

```
#import "SceneDelegate.h"
#import <UserNotifications/UserNotifications.h>
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TIMDefine.h"

@interface SceneDelegate ()
@property (nonatomic, strong) NSString *roomId;
@end

@implementation SceneDelegate

- (void)scene:(UIScene *)scene willConnectToSession:(UISceneSession
*)session options:(UISceneConnectionOptions *)connectionOptions {
    [self processOfflinePush:connectionOptions];
}

- (void)processOfflinePush: (UISceneConnectionOptions
*)connectionOptions {
    NSDictionary *pushNotification =
connectionOptions.notificationResponse.notification.request.content.
userInfo;
    NSString *notice = pushNotification[@"ext"];
    NSDictionary *dic = [self dictionaryFromString:notice];
    NSString *roomId = dic[@"RoomId"];
    //将roomId传给您自己的登录页面YourLoginViewController
}

@end
```

在您的登录页面完成 TUICore 的登录，并且判断是否需要跳转到会议主界面。

Swift

```
import TUICore
import TUIRoomKit

//YourLoginViewController 是您自己的登录页面
class YourLoginViewController: UIViewController {
    var roomId: String?

    override func viewDidLoad() {
        super.viewDidLoad()
        TUILogin.login(Int32(SDKAppID), userID: "yourUserName",
userSig: "yourUserSig") { [weak self] in
            guard let self = self else { return }

self.navigationController?.pushViewController(YourSelfViewController
(), animated: false)
            //YourSelfViewController是正常登录后应该显示的您自己的界面
            guard let roomId = self.roomId else { return }
            //通过roomId判断是否通过离线通知进入前台，由此判断是否再跳转到会议
主界面
            self.showConferenceMainViewController(roomId: roomId)
            self.roomId = nil
        } fail: { (code, errorDes) in
            print("code:\(code), errorDes:\(String(describing:
errorDes))")
        }
    }

    func showConferenceMainViewController(roomId: String) {
        let conferenceViewController =
ConferenceMainViewController()
        let params = JoinConferenceParams(roomId: roomId)
        conferenceViewController.setJoinConferenceParams(params:
params)

navigationController?.pushViewController(conferenceViewController,
animated: true)
    }
}
```

OC

```
#import "YourLoginViewController.h"
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TUILogin.h"

@interface YourLoginViewController ()
@property (nonatomic, strong) NSString *roomId;
@end

@implementation YourLoginViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [TUILogin login:yourSDKAPPID userID:@"youruserID"
userSig:@"yourUserSig" succ:^{
        //先显示您自己的界面
        //如果roomId有值，说明是从离线推送进来的，可以调用
showConferenceMainViewController再跳转到会议主界面
    } fail:^(int code, NSString * _Nullable msg) {
    }];
    // Do any additional setup after loading the view.
}

- (void)showConferenceMainViewController: (NSString *)roomId {
    ConferenceMainViewController * vc =
[[ConferenceMainViewController alloc]init];
    JoinConferenceParams * params = [[JoinConferenceParams
alloc]initWithRoomId:
                                     roomId isOpenMicrophone:true
isOpenCamera:false isOpenSpeaker:true];
    [vc setJoinConferenceParamsWithParams:params];
    [self.navigationController pushViewController:vc animated:true];
}
```

如果是从后台进入前台，需要在 AppDelegate文件中实现 `onRemoteNotificationReceived` 方法。

Swift

```
import TUIRoomKit
import TIMPush

@main
class AppDelegate: UIResponder, UIApplicationDelegate,
TIMPushDelegate {
    var roomId: String?
    func onRemoteNotificationReceived(_ notice: String?) -> Bool {
        guard let notice = notice else { return false }
        guard let dict = convertToDic(string: notice) else { return
false }
        guard let roomId = dict["RoomId"] as? String else { return
false }
        if V2TIMManager.sharedInstance().getLoginStatus() ==
.STATUS_LOGINED {
            //如果从后台唤醒并且已经完成登录，可以直接进入会议。
            showConferenceMainViewController(roomId: roomId)
        }
        return true
    }
}
```

OC

```
#import "AppDelegate.h"
#import "TIMPush/TIMPushManager.h"
#import "TUIRoomKit/TUIRoomKit-Swift.h"
#import "TIMDefine.h"

@interface AppDelegate ()<TIMPushDelegate>
@property (nonatomic, strong) NSString *roomId;
@end
@implementation AppDelegate

- (BOOL)onRemoteNotificationReceived:(NSString *)notice {
    NSDictionary * dic = [self dictionaryFromString:notice];
    NSString * roomId = dic[@"RoomId"];
    if (!roomId) {
        return false;
    }
    if ([V2TIMManager sharedInstance].getLoginStatus ==
V2TIM_STATUS_LOGINED ) {
```

```
            //如果从后台唤醒并且已经完成登录，可以直接进入会议。
            [self showConferenceMainViewController:roomId];
        }
        return true;
    }

    @end
```

常见问题

1. 若在集成过程中遇到问题，请务必先查阅 插件推送-常见问题 进行自助排查。
2. 条件说明：**部分厂商要求必须上架应用市场**才可以正常使用推送服务，详情参见下表：

厂商通道	是否需要上架	账号说明
小米	是	需要注册企业开发者账号
VIVO	是	需要注册企业开发者账号
OPPO	否	需要注册企业开发者账号
荣耀	否	需要注册企业开发者账号
华为	否	个人开发者账号即可
魅族	否	个人开发者账号即可

交流与反馈

如有问题，欢迎您加入我们的 TUIRoomKit 技术交流平台 zhiliao ，进行技术交流和产品沟通。

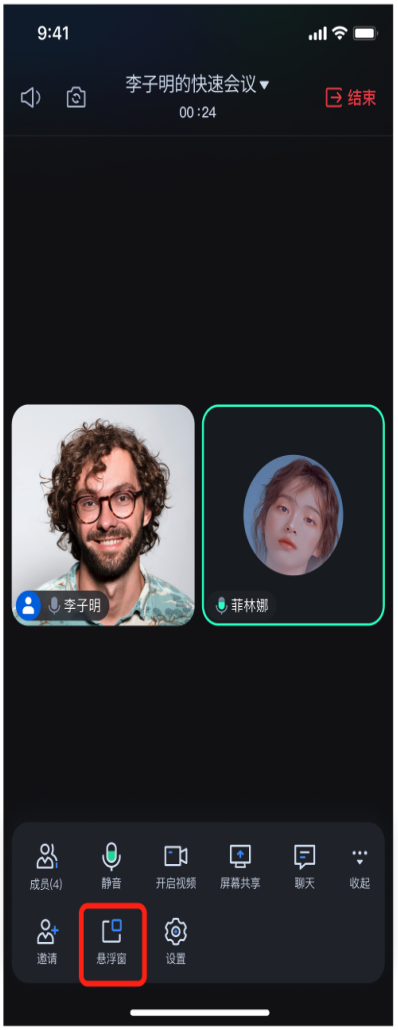
悬浮窗

最近更新时间：2024-05-17 14:44:23

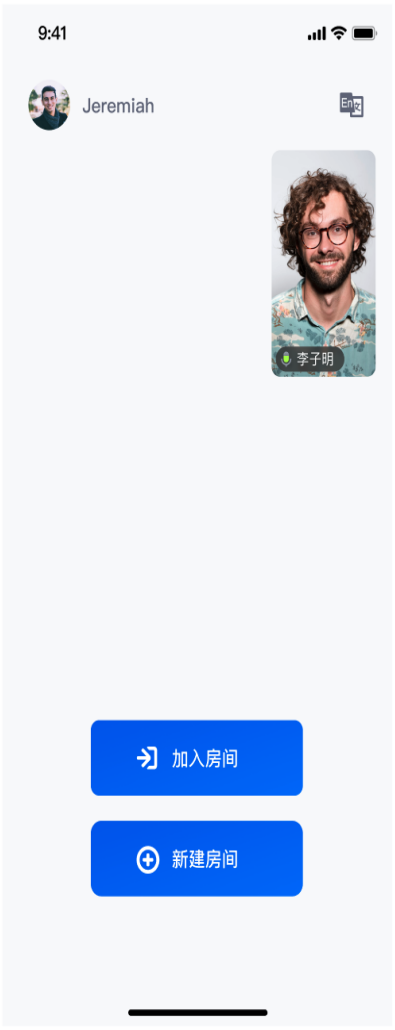
功能介绍

TUIRoomKit 支持悬浮窗功能，允许用户创建一个可自由拖动的悬浮窗口，用于展示和管理 TUIRoomKit 的视频会议界面，使用户在参与视频会议时能够更轻松地同时处理其他任务。

悬浮窗按钮



悬浮窗样式



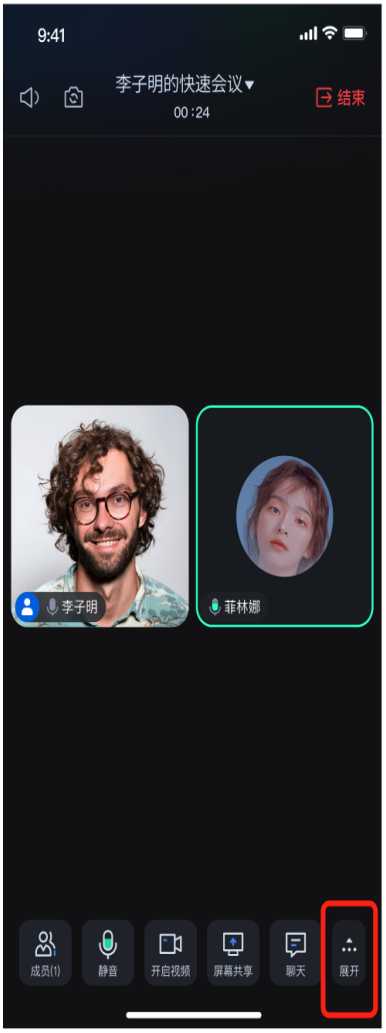
支持平台

- Android。
- iOS。

操作流程

1. 会中点击底部工具栏**展开** > **悬浮窗**即可开启悬浮窗。
2. 首次开启**悬浮窗**功能时，Android 机型会跳转相关系统设置页面，需勾选开启应用的相关权限，如**悬浮窗**、**后台弹出界面**、**允许显示在其他应用的上层**等。不同机型的相关系统设置项名称可能略有不同，因此您需要根据具体机型进行相应设置。

底部栏 -> 展开按钮



Android系统设置示例

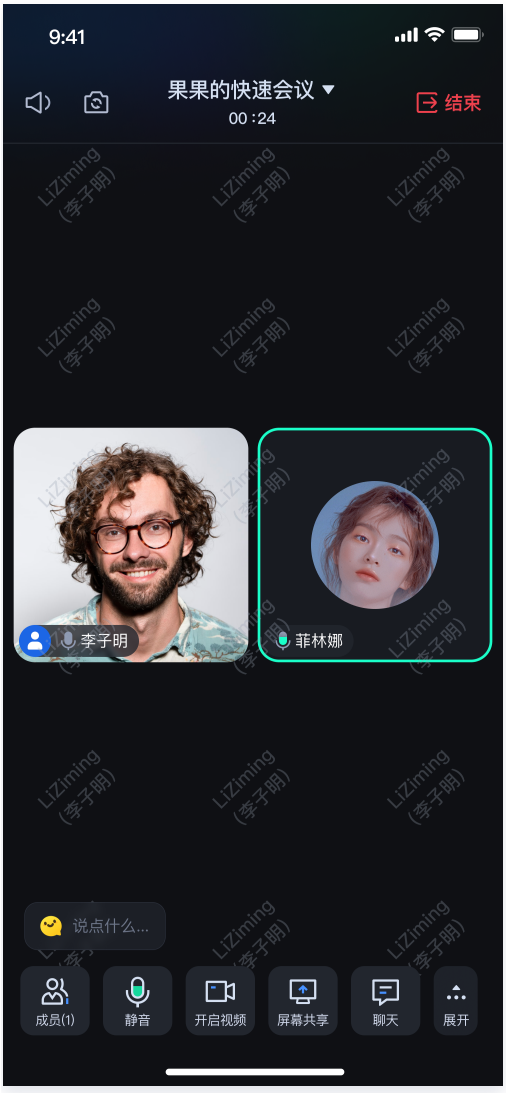
3. 在开启系统相关权限后，Android 端支持应用内和应用外的悬浮窗，iOS 端仅支持应用内的悬浮窗。
4. 在悬浮窗状态下，点击悬浮窗即可返回会议。

文字水印
Android&iOS

最近更新时间：2024-09-26 11:07:11

功能介绍

TUIRoomKit 支持文字水印功能，用户可以在会议中添加自定义的文字水印，以保护内容版权或传达特定信息。通过文字水印功能，用户可以在会议画面上显示个人信息、公司名称或会议主题等信息，增强内容的安全性和专业性。本文将详细介绍该特性的相关功能，并说明如何在 TUIRoomKit 组件中使用这一特性。



功能接入

开启水印

TUIRoomKit中，水印功能默认关闭。如果您想开启文字水印功能，可以通过以下代码进行开启。

Android

```
ConferenceSession.sharedInstance().enableWaterMark();
```

iOS

```
ConferenceSession.sharedInstance.enableWaterMark()
```

说明：
开启后，默认水印的文本内容为 您的userId(您的userName) 。

设置水印文本

在 TUIRoomKit 中，您可以自定义水印显示的文本内容以满足您特定的业务需求。您可以通过如下代码设置您的水印文本内容。

Android

```
ConferenceSession.sharedInstance().setWaterMarkText("yourWaterMarkTe
xt");  // 将字符串替换为您需要设置的水印内容
```

iOS

```
ConferenceSession.sharedInstance.setWaterMarkText(waterMarkText:
"yourWaterMarkText")  // 将字符串替换为您需要设置的水印内容
```

功能定制

如果当前的 UI 不满足您的需求，您可以通过修改源代码来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对文字水印相关的文件做了介绍。

Android

您可以通过修改 Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/WaterMark 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对文字水印相关的文件做了介绍。

```
// 文件位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/vi
ew/page/widget/WaterMark

WaterMark                              // 文字水印相关的视图目录
    ├── TextWaterMarkView.java         // 文字水印视图
    └── WaterMarkLineStyle.java        // 文字水印格式
```

iOS

您可以通过修改 iOS/TUIRoomKit/Source/View/Page/Widget/WaterMark 目录下的源代码，来实现您满意的 UI 效果。为了您更方便的定制 UI，这里对文字水印相关的文件做了介绍。

```
// 文件位置：iOS/TUIRoomKit/Source/View/Page/Widget/WaterMark

WaterMark                              // 文字水印相关的视图目录
    ├── WaterMarkLayer.swift           // 文字水印视图
    └── WaterMarkLineStyle.swift       // 文字水印格式
```

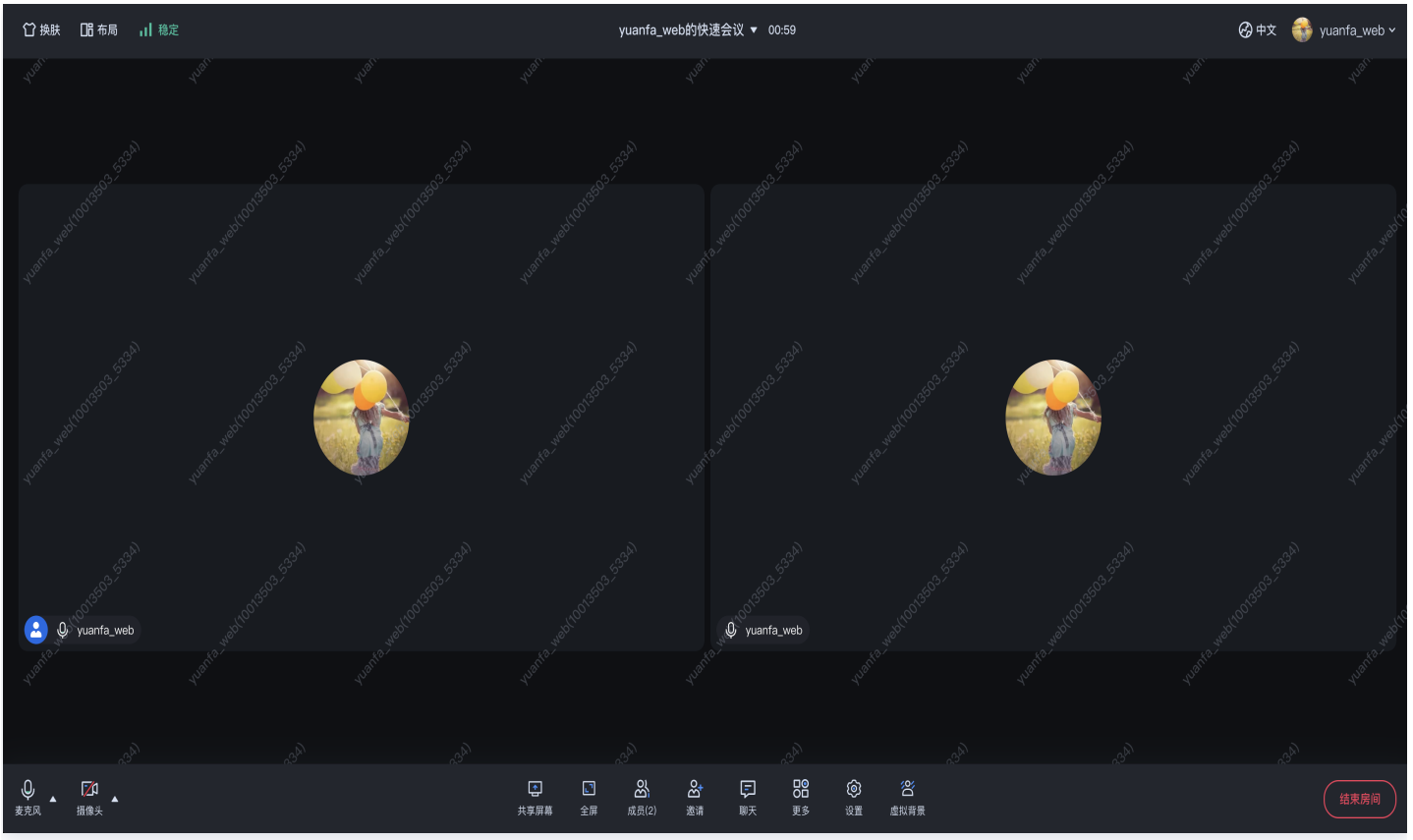
Web&Electron

最近更新时间：2024-05-24 15:19:21

TUIRoomKit 推出了文字水印功能，让用户在进行多人会议时可以设置文字水印。 本文将详细介绍如何在 TUIRoomKit 组件中使用这一特性。

集成效果

在 TUIRoomKit 组件中集成文字水印功能后，显示效果如下：



准备条件

在使用腾讯云提供的文字水印功能前，您需要前往控制台，为应用开通多人会议服务。具体步骤请参见 开通服务 。

开启文字水印

注意：

- 小程序平台暂不支持，electron 与 web 端均已支持。
- 需使用 TUIRoomKit v2.3.3 及以上版本。

TUIRoomKit 提供了设置文字水印的功能。您可以通过调用 enableWatermark 接口来启用文字水印功能。

Web

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-web-vue2.7
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableWatermark();
```

Electron

```
// 注意包的名称，如果您使用 vue2 版本请更改包名为 @tencentcloud/roomkit-electron-vue2.7
import { conference } from '@tencentcloud/roomkit-electron-vue3';
conference.enableWatermark();
```

常见问题

开启文字水印后页面样式显示异常?

请检查 id 为 'roomContainer' 的父级元素，确认其 `style` 是否已设置为：`width: 100%; height: 100%; overflow: hidden;` 。

虚拟背景

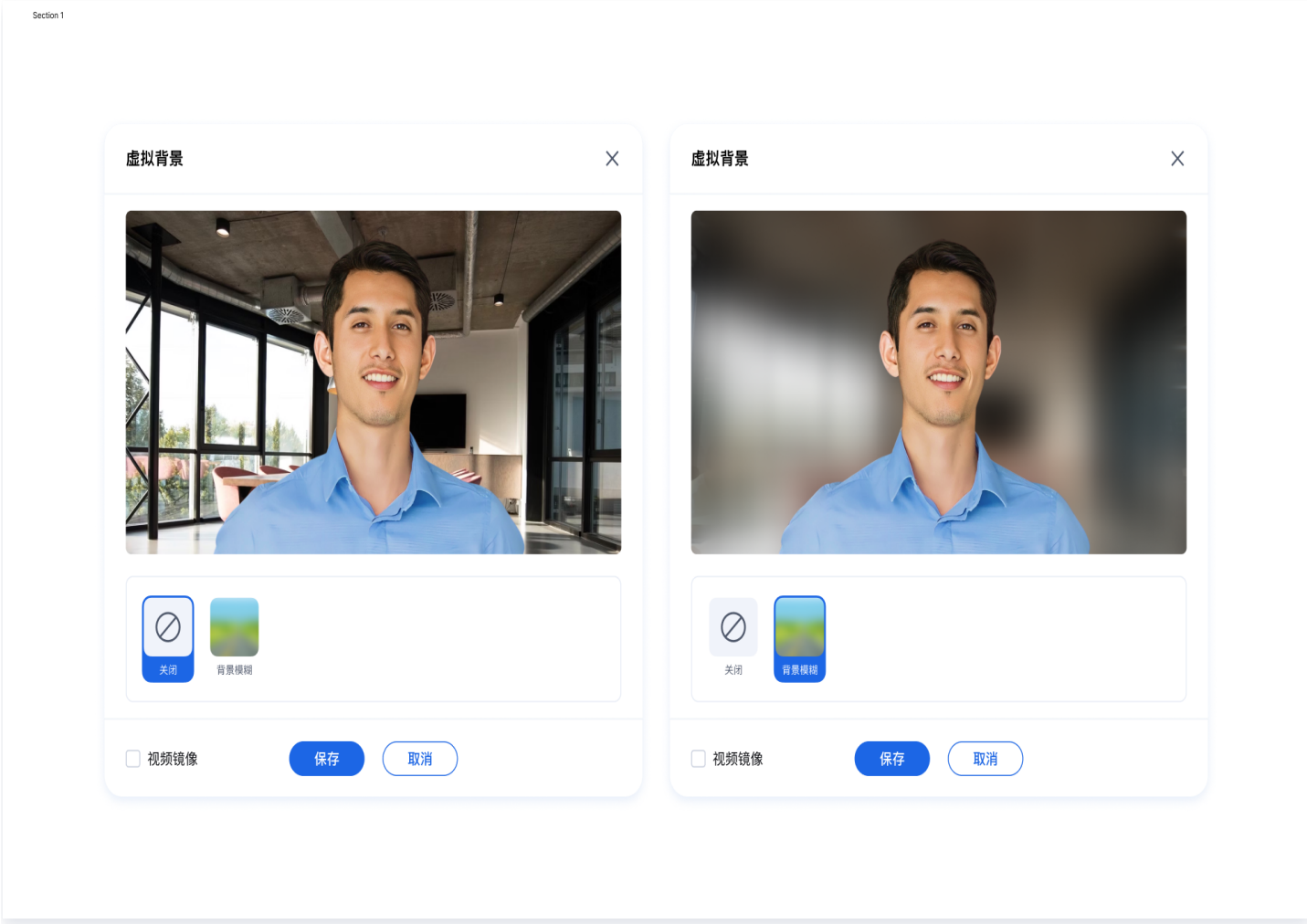
Web

最近更新时间：2024-05-24 15:19:21

TUIRoomKit 推出了虚拟背景功能，让用户在进行多人会议时可以设置模糊背景，隐藏实际会议环境，保护隐私并增加会议趣味性。 本文将详细介绍如何在 TUIRoomKit 组件中使用这一特性。

集成效果

在 TUIRoomKit 组件中集成虚拟背景功能后，显示效果如下：



准备条件

在使用腾讯云提供的虚拟背景功能前，您需要前往控制台，为应用开通多人会议服务，并购买 **进阶互动版/超大房间互动版** 套餐。具体步骤请参见 开通服务 。

开启模糊背景

> ⚠ **注意：**
> - **H5 暂不支持，仅适用于 Web PC 端。**
> - **需使用 TUIRoomKit v2.3.3 及以上版本。**

TUIRoomKit 的 UI 方案支持设置虚拟背景。您可以通过调用 enableVirtualBackground 接口，在 UI 上显示虚拟背景功能按钮。点击该按钮即可启用虚拟背景功能。

Vue3

```
import { conference } from '@tencentcloud/roomkit-web-vue3';
conference.enableVirtualBackground();
```

Vue2

```
import { conference } from '@tencentcloud/roomkit-web-vue2.7';
conference.enableVirtualBackground();
```

常见问题

开启虚拟背景无反应或有延迟?

- 确保已购买多人会议 **进阶互动版/超大房间互动版** 套餐，详见 开通服务 。
- 网络较差时，虚拟背景模型文件可能未下载完，从而导致开启虚拟背景失败。

关闭摄像头，是否能开启虚拟背景?

不能。

监听会议状态

最近更新时间：2024-08-29 17:20:11

本文介绍 TUIRoomKit 组件会议状态回调的使用。

会议状态监听

如果您的业务需要**监听会议的状态**，例如会议开始、结束等会议过程中的事件，可以参见如下代码：

Android（Java）

```
ConferenceDefine.ConferenceObserver observer = new
ConferenceDefine.ConferenceObserver() {
    @Override
    public void onConferenceStarted(TUIRoomDefine.RoomInfo roomInfo,
TUICommonDefine.Error error, String message) {
    }

    @Override
    public void onConferenceJoined(TUIRoomDefine.RoomInfo roomInfo,
TUICommonDefine.Error error, String message) {
    }

    @Override
    public void onConferenceExisted(String roomId) {
    }

    @Override
    public void onConferenceFinished(String roomId) {
    }
};
ConferenceSession.sharedInstance().addObserver(observer);
```

Android（Kotlin）

```
val observer: ConferenceObserver = object : ConferenceObserver() {
    override fun onConferenceStarted(roomInfo: TUIRoomDefine.RoomInfo?,
error: TUICommonDefine.Error?, message: String?) {
    }

    override fun onConferenceJoined(roomInfo: TUIRoomDefine.RoomInfo?,
```

```
error: TUICommonDefine.Error?, message: String?) {
    }

    override fun onConferenceExisted(roomId: String?) {
    }

    override fun onConferenceFinished(roomId: String?) {
    }
}
ConferenceSession.sharedInstance().addObserver(observer)
```

iOS (Swift)

```
import TUIRoomKit

class EnterRoomViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        ConferenceSession.sharedInstance.addObserver(observer: self)
    }
}


extension EnterRoomViewController: ConferenceObserver {
    // 发起会议回调
    func onConferenceStarted(roomInfo: TUIRoomInfo, error: TUIError,
message: String) {
        // Your code here
     }

    // 加入会议回调
    func onConferenceJoined(roomInfo: TUIRoomInfo, error: TUIError,
message: String) {
        // Your code here
    }

    // 会议被解散回调
    func onConferenceFinished(conferenceId: String) {
        // Your code here
    }
```

```
    // 退出会议回调
    func onConferenceExited(conferenceId: String) {
        // Your code here
    }
}
```

iOS (OC)

```
@interface EnterRoomViewController () <ConferenceObserver>

@end

@implementation EnterRoomViewController
- (void)viewDidLoad {
    [super viewDidLoad];
    [[ConferenceSession sharedInstance] addObserver:self];
 }

#pragma mark - ConferenceObserver
// 发起会议回调
- (void)onConferenceStartedWithRoomInfo:(TUIRoomInfo *)roomInfo error:
(TUIError *)error message:(NSString *)message {
    // Your code here
}

// 加入会议回调
- (void)onConferenceJoinedWithRoomInfo:(TUIRoomInfo *)roomInfo error:
(TUIError *)error message:(NSString *)message {
    // Your code here
}

// 会议被解散回调
- (void)onConferenceFinishedWithConferenceId:(NSString *)conferenceId {
    // Your code here
}

// 退出会议回调
- (void)onConferenceExitedWithConferenceId:(NSString *)conferenceId {
    // Your code here
}

@end
```