多人音视频房间 SDK

预定会议

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【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

预定会议
Android&iOS&Flutter
Web&Electron

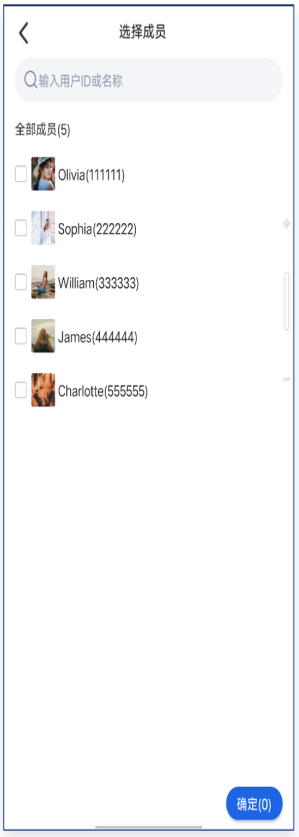
预定会议
Android&iOS&Flutter

最近更新时间：2025-05-07 09:38:11

若 TUIRoomKit 的 UI 交互不满足您的产品需求，您有自己的交互和业务逻辑需要自定义实现预定会议相关的交互功能，可以接入 TUIRoomEngineSDK，并参考 关键代码 相关调用来实现您的需求。

功能介绍

TUIRoomKit 支持预定房间，用户可以预定一个房间并安排日程中的会议。当会议时间到来时，用户只需一键即可开启会议沟通。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。

预定会议	会议列表	设置会议	邀请成员
			

功能接入

在使用 TUIRoomKit 提供的预定房间特性之前，您需要完成 TUIRoomKit 组件接入的相关配置及登录，详情可参见 快速接入 。

> **注意：**
> 预定房间特性需使用 TUIRoomKit v2.5.0 及以上版本。

如何预定房间

使用预定房间功能时，您需要拉起 TUIRoomKit 中提供的预定房间页面，不同平台请参考：

Android

根据您的业务，只需在对应的 Activity 中调用如下代码即可预定一个房间。

java

```java
Intent intent = new Intent(this, ScheduleConferenceActivity.class);
startActivity(intent);
```

kotlin

```kotlin
val intent = Intent(this, ScheduleConferenceActivity::class.java)
startActivity(intent)
```

iOS

```ios
class YourViewController: UIViewContrller {
    func jumpToScheduleViewController {
        let scheduleViewController =
ScheduleConferenceViewController()

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

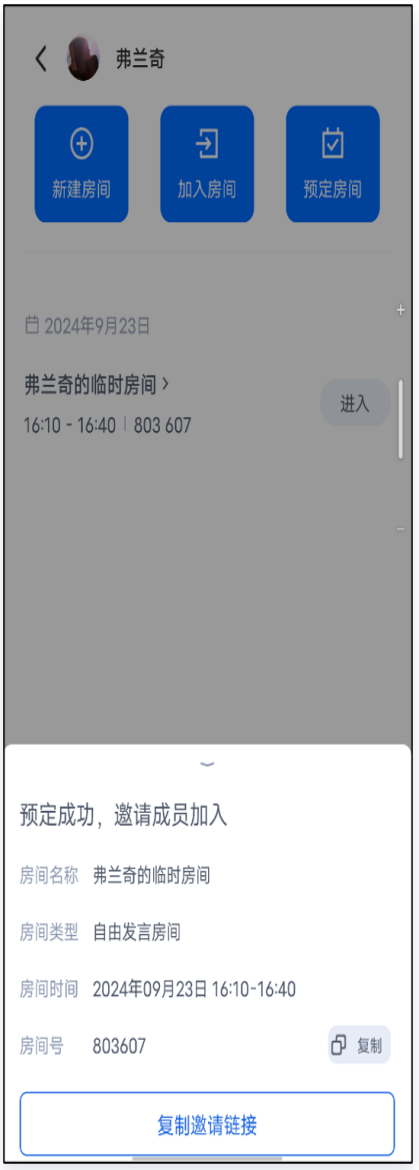
Flutter

根据您的业务，在您需要时自行导航至 `ScheduleRoomPage` 预定房间界面。

```
import
'package:tencent_conference_uikit/tencent_conference_uikit.dart

Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => ScheduleRoomPage(),
  ),
);
```

- 预定会议可设置的属性：其中可设置内容包含：房间名称、房间类型、开始时间、房间时长、时区、邀请成员（需导入成员列表）、房间加密、成员管理（全体静音、全体禁画）等。
- 如何邀请成员：点击 "添加" 即可邀请成员。**默认情况下用户列表需要您传入，为了方便您快速体验，我们提供了用户列表 Demo。**具体请参考 如何邀请成员 。

进入预定会议界面	设置会议信息	预定成功



查看已预定的房间

TUIRoomKit 中提供了房间列表的UI组件。通过将房间列表组件自行布局到自己的页面上，您便可以完成对当前用户所有已预定房间的查看与修改：

Android

1. 根据您的业务，在您的布局中添加会议列表的布局：

```
<!-- 以您的父布局是ConstraintLayout为例，添加如下代码 -->
<FrameLayout
    android:id="@+id/tuiroomkit_fl_conference_list_container"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"/>
```

2. 在对应的 Activity 中添加如下代码，拉起会议列表：

java

```
ConferenceListFragment fragment = new ConferenceListFragment();
FragmentTransaction transaction =
this.getSupportFragmentManager().beginTransaction();
transaction.add(R.id.tuiroomkit_fl_conference_list_container,
fragment);
transaction.commitAllowingStateLoss();
```

kotlin

```
val fragment = ConferenceListFragment()
val transaction = supportFragmentManager.beginTransaction()
transaction.add(R.id.tuiroomkit_fl_conference_list_container,
fragment)
transaction.commitAllowingStateLoss()
```

iOS

```
class YourViewController: UIView {
        // ConferenceListView初始化时需要两个参数:
        // @param viewController, 会议列表页被添加到的viewcontroller
        func showConferenceList {
             let listView = ConferenceListView(viewController: self)
             view.addSubview(listView)
        }
```

```
}
```

Flutter

```
import 'package:flutter/material.dart';
import
'package:tencent_conference_uikit/tencent_conference_uikit.dart';

// 在您自己的页面中，添加ConferenceListWidget
class YourPage extends StatelessWidget {
  const YourPage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Your Page")),
      body: Column(
        children: [
          ...YourWidgets(),              // 您其他的widget，这里仅作示
例。
          const ConferenceListWidget(),  // 会议列表组件
        ],
      ),
    );
  }
}
```

会议列表提供以下功能：

- **查看会议列表：列表中包括您创建的会议以及您被邀请参加的会议。**
- **查看会议详情：您可以点击某个会议查看该会议的详情。**
- **修改会议信息：点击会议列表某个会议，若该会议未进行，且您是发起人，您可以修改该会议信息。**

点击会议列表	查看该会议详情	修改会议信息

如何邀请成员

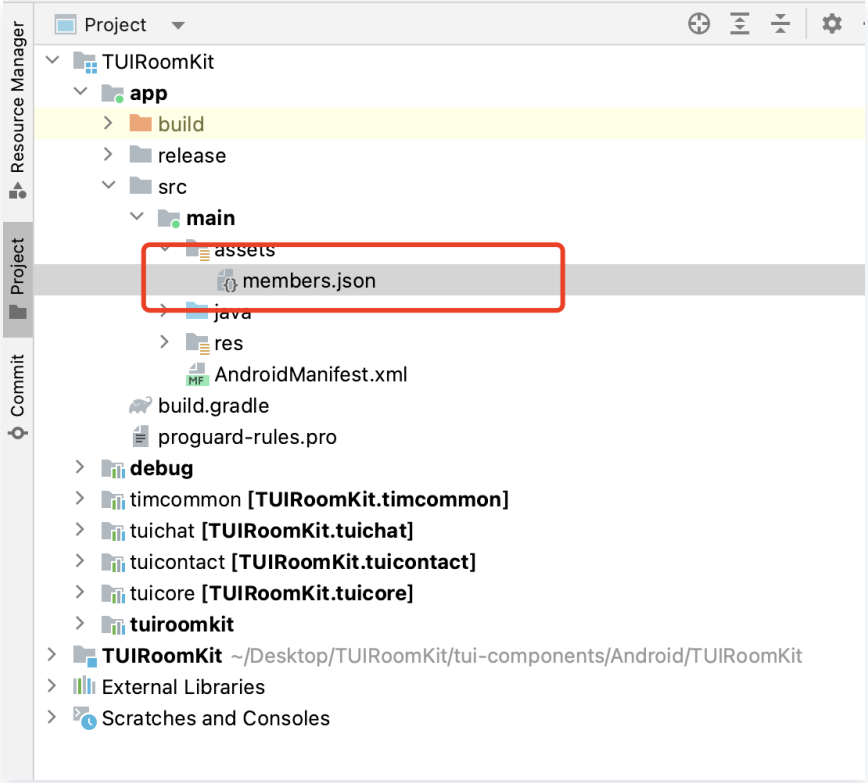
您可在预定房间界面或修改房间界面，通过点击 参与成员 按钮弹出成员选择界面。**默认情况下用户列表需要您传入（您可以使用我们提供用户列表Demo快速体验），您可以使用如下方式，根据您的业务需求导入需要邀请的成员，**不同平台请参考：

Android

如何体验邀请成员的功能

首先参考 跑通 Demo，我们在 Demo 工程中提供了用户列表（Android/app/src/main/java/com/tencent/liteav/demo/SelectParticipants），并在

`members.json` 文件中已经预先配置了一些用于测试的用户信息，可以选取两个账号，分别在两个手机用我们配置的 userId 登录，然后在预定会议或者编辑会议时选择另一个用户，这样另一个用户的预定列表中就会出现刚才预定的会议。



如何使用自定义通讯录

1. **TUIRoomKit 关联自定义通讯录：**您可在 预定会议 和 拉起会议列表 前，通过以下方法设置自定义通讯录：

java

```
// 将 SelectParticipantActivity.class 替换为自定义通讯录的activity
ConferenceSession.sharedInstance().setContactsViewProvider(SelectPar
ticipantActivity.class);
```

kotlin

```
// 将 SelectParticipantActivity::class.java 替换为自定义通讯录的
activity
ConferenceSession.sharedInstance().setContactsViewProvider(SelectPar
ticipantActivity::class.java)
```

> **说明：**
> `SelectParticipantActivity` 为自定义通讯录代码示例，您可在 Demo 工程下(目录：app/src/main/java/com/tencent/liteav/demo/SelectParticipants) 查看。

2. **自定义通讯录向 TUIRoomKit 返回选择完毕的用户名单：**在通讯录完成用户选择后，您需要将已选用户列表返回给 TUIRoomKit。您可以通过以下方法将数据返回给 TUIRoomKit。

java

```java
Intent intent = new Intent();
// participants 为选择完毕的用户列表，必须为 ArrayList<User> 类型。
intent.putExtra(SELECTED_PARTICIPANTS, participants);
setResult(3, intent);
finish();
```

kotlin

```kotlin
val intent = Intent()
// participants 为选择完毕的用户列表，必须为 ArrayList<User> 类型。
intent.putExtra(SELECTED_PARTICIPANTS, participants)
setResult(3, intent)
finish()
```

3. **修改会议成员时 TUIRoomKit 向通讯录传递已选择的成员名单：**您可通过以下方法获得该会议已选择的用户名单。

java

```java
// 获取已选择的用户列表
ConferenceParticipants participants = (ConferenceParticipants)
bundle.getSerializable(CONFERENCE_PARTICIPANTS);
ArrayList<User> selectedList = participants.selectedList;
```

kotlin

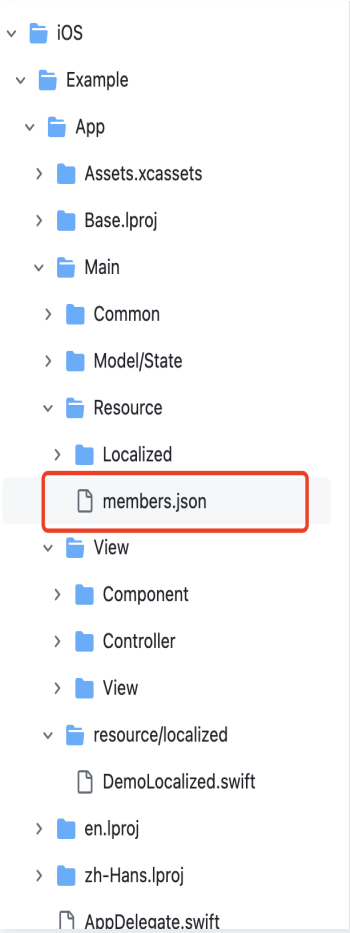
```kotlin
// 获取已选择的用户列表
val participants = bundle.getSerializable(CONFERENCE_PARTICIPANTS)
as ConferenceParticipants
```

```
val selectedList: ArrayList<User> = participants.selectedList
```

iOS

如何体验邀请成员的功能

首先参考 跑通 Demo，完成 demo 跑通。在 demo 工程的 `members.json` 文件中已经预先配置了一些用于测试的用户信息，可以选取两个账号，分别在两个手机用我们配置的 userId 登录，然后在预定会议或者编辑会议时选择另一个用户，这样另一个用户的预定列表中就会出现刚才预定的会议。




```
[
    {
        "userId": "111111",
        "userName": "Olivia",
        "avatarUrl": "https://liteav.sdk.qcloud.com/app/res/picture/voiceroom/avatar/user_avatar1.png"
    },
    {
        "userId": "222222",
        "userName": "Sophia",
        "avatarUrl": "https://liteav.sdk.qcloud.com/app/res/picture/voiceroom/avatar/user_avatar1.png"
    },
    {
        "userId": "333333",
        "userName": "William",
        "avatarUrl": "https://liteav.sdk.qcloud.com/app/res/picture/voiceroom/avatar/user_avatar1.png"
    },
    {
        "userId": "444444",
        "userName": "James",
        "avatarUrl": "https://liteav.sdk.qcloud.com/app/res/picture/voiceroom/avatar/user_avatar1.png"
    },
    {
        "userId": "555555",
        "userName": "Charlotte",
        "avatarUrl": "https://liteav.sdk.qcloud.com/app/res/picture/voiceroom/avatar/user_avatar1.png"
    }
]
```

代码实现成员列表页

考虑到邀请成员页的用户列表数据的复杂性，我们设计了允许您自定义成员选择界面的方案，接下来将指导您如何集成自己的成员选择页（当然您也可以直接使用我们在 demo 中提供的 UI，这个将在后面介绍）。

1. 准备好您的好友选择页 viewController，实现 `ContactViewProtocol` 协议。

```
// 示例代码
class SelectMemberViewController: UIViewController,
ContactViewProtocol {
    weak var delegate: ContactViewSelectDelegate?
    var selectedUsers: [User]

    func didSelectFinished() {
    // 在完成选择的方法中通过delegate把选择的成员回调给RoomKit
        delegate?.onMemberSelected(self, invitees:
selectedMembers)
    }
}
```

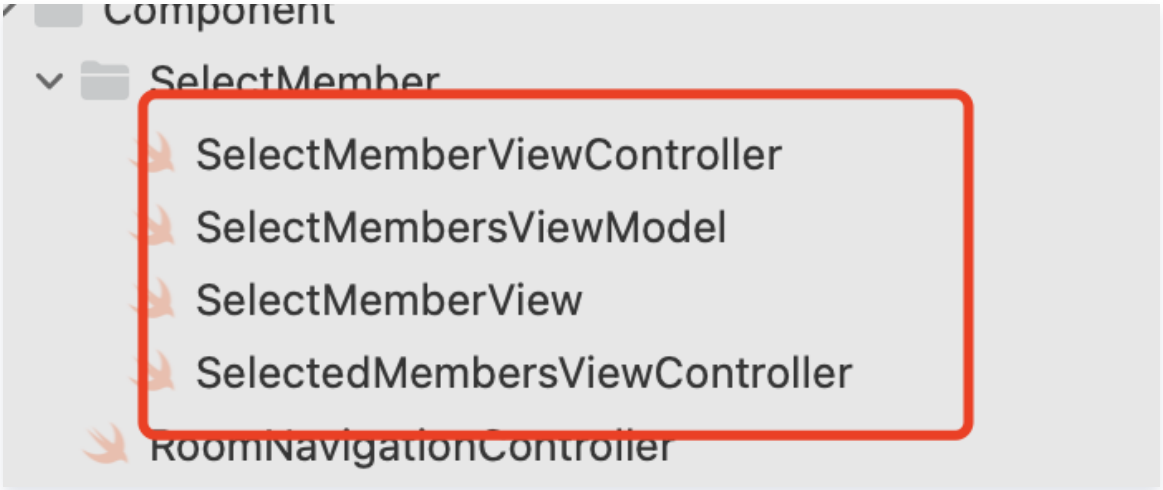
说明：
会中呼叫功能同样需要使用您自定义的通讯录组件。如您还需使用会中呼叫功能，这里建议您将一个 `ConferenceParticipants` 对象置于您通讯录页面的构造函数参数中，数据来源会在第二步的代码中提到。
`ConferenceParticipants` 类中有两个成员：

- selectedList：已选择的成员；
- unSelectableList：不可选择的成员，您可以在UI上将对应的成员设置其为不可选择。预定会议功能没有使用此字段。

2. **TUIRoomKit 关联自定义通讯录：您可在预定会议和拉起会议列表前，通过以下方法设置您在上一步中自定义的通讯录：**

```
// 示例代码
ConferenceSession.sharedInstance.setContactsViewProvider {
participants in
    return SelectMemberViewController(participants: participants)
}
```

3. 通过以上两步就可以展示您自己的成员选择页了，同时我们在 demo 中提供了上面图片示例的页面代码，您可以直接把以下几个文件拷贝您的工程中，就可以直接获得我们的示例页面。



在 `SelectMembersViewModel` 的 `loadMembers` 方法中您可以加载自己的成员列表数据（也可以直接获取 IM 关系链数据）。

Flutter

当您需要使用邀请成员进入房间的功能时，我们提供了以下两种方案供您添加需要邀请的成员：

方案一：使用 json 配置用户信息

您可以参考我们的 example 工程，在 example/assets 目录下， `members.json` 中存储了邀请所需要的用户信息。使用步骤如下：

1. 在您项目的 `assets` 目录下，新增 `members.json` 文件，列出所有您所需要的用户信息，文件格式与上文提到的 `members.json` 相同。
2. 在您项目的 `pubspec.yaml` 中，完成以下配置：

```
assets:
  - assets/members.json
```

完成以上配置后，您便可以在邀请成员界面中选择您在 `members.json` 中所列出的成员。

方案二：使用 IM 好友信息

当您没有配置上文所描述的 `members.json` 时，邀请好友界面会默认拉取您的IM好友信息。当您需要邀请其他成员进入房间前，您可以根据您的业务侧需求，自行添加想要邀请的成员为好友。

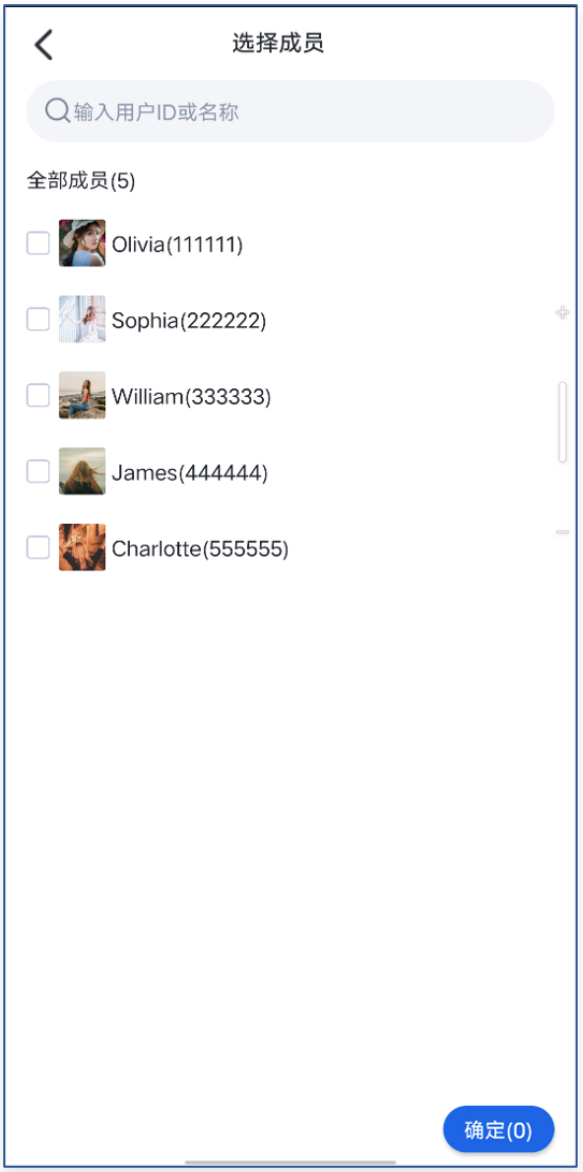
- 前提条件：登录 IM SDK，登录操作与 登录弹幕聊天 操作相同。如您已完成其登录或已使用 会中聊天，可忽略此步骤。
- 添加好友的代码如下：

```
import
'package:tencent_cloud_chat_sdk/manager/v2_tim_manager.dart';

// Flutter TUIRoomKit中，已有对tencent_cloud_chat_sdk的依赖，您无需另
行配置
// 将代码中的userID替换为您需要添加的用户的UserID即可完成好友的添加
V2TIMManager().getFriendshipManager().addFriend(
        userID: 'userID', addType:
FriendTypeEnum.V2TIM_FRIEND_TYPE_BOTH);
```

完成好友的添加后，您便可以选择在**每次**预定时完成对所添加用户的邀请。
如您需要更多相关的好友操作，请参考 好友管理 。如您需要使用Rest API批量添加好友，请参考 添加好友RestAPI 。

完成以上内容后，通过点击 参与成员 按钮弹出成员选择界面，您可以在这里邀请或删除参与的成员。邀请并完成房间的预定或修改后，对方的会议列表中便会**出现您所预定的会议**。

预定会议界面 点击 参与成员	用户列表中添加或删除成员



注意事项

- 预定房间成功后，即可获取房间号和预定信息。
- 若同时预定不同日期/时间的房间，分次选择时间提交即可。
- 预定房间开始时间不能早于当前时间，但没有提前天数的限制。
- 预定的房间到达结束时间后，若房间没有被解散且房间内没有用户，该房间将从预定结束时间起保留6小时，在此期间您仍可以随时进入该房间。

功能定制

如果当前的 UI 不满足您的需求，您可以通过修改源码来实现您满意的 UI 效果，不同平台请参考：

Android

您可以通过修改 Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/schedule 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，这里对房间密码的文件做了介绍。

```
SchdeuleConference
    ├── ConferenceDetails          // 会议详情
    ├── ConferenceList             // 会议列表
    ├── ModifyConference           // 修改会议
    ├── SelectScheduleParticipant  // 邀请成员
    ├── TimeZone                   // 设置时区
    └── View                       // 预定会议
```

iOS

您可以通过修改 iOS/TUIRoomKit/Source 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，这里对房间密码的文件做了介绍。

```
Source
├── ConferenceListView.swift                // 预定会议列表页面
├── ScheduleConferenceViewController.swift  // 预定会议主页面
└── View
    └── ConferenceOptions
        ├── ConferenceInvitation            // 会中呼叫
        ├── MemberSelect                    // 预定会议邀请成员
        ├── ModifySchedule                  // 修改预定会议信息
        └── ScheduleConference
            └── View
                ├── ScheduleConference      // 预定会议
                ├── ScheduleDetails         // 预定会议详情
                └── Widget                  // 预定会议弹出页面
```

Flutter

您可以通过修改 tencent_conference_uikit/lib/pages 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，这里对预定会议的文件做了介绍。

```
pages
  ├── conferenceList                          // 预定会议列表文件夹
  |       ├── view.dart                       // 预定会议列表页面
  |       └── widget.dart
  |           ├── conference_date_item.dart   // 会议列表中单个时间
widget
  |           ├── conference_item.dart        // 会议列表中单个会议
widget
  |           └── no_schedule_widget.dart     // 会议列表中暂无会议时
显示widget
  ├── schedule_conference                     // 预定会议页面文件夹
  |       ├── view.dart                       // 预定会议页面
  |       └── widgets
  |           ├── attendee_selector           // 成员选择页面文件夹
  |           |       ├── view.dart           // 成员选择页面
  |           |       └── widgets
  |           |           └── selected_attendees.dart  // 弹出的已
选择成员页面
  |           ├── duration_selector.dart      // 预定会议时长选择
widget
  |           ├── room_control.dart           // 预定会议全体静音/禁画
widget
  |           ├── room_info.dart              // 预定会议总体房间信息
widget
  |           ├── room_type.dart              // 房间类型选择widget
  |           ├── start_time_selector.dart    // 预定会议开始时间选择
widget
  |           └── time_zone_selector.dart     // 预定会议时区选择页
  └── schedule_details                        // 会议详情文件夹
  |       ├── view.dart                       // 会议详情页面
  |       └── widgets
  |           ├── details_button_item.dart    // 会议详情页面内单个按
钮widget
  └           └── room_info.dart              // 会议信息widget
```

说明：

> 如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。

关键代码

若您想自定义实现预定会议功能，请参考 TUIConferenceListManager： Android 、iOS&Mac 、Flutter 。

Web&Electron

最近更新时间：2024-08-08 17:00:11

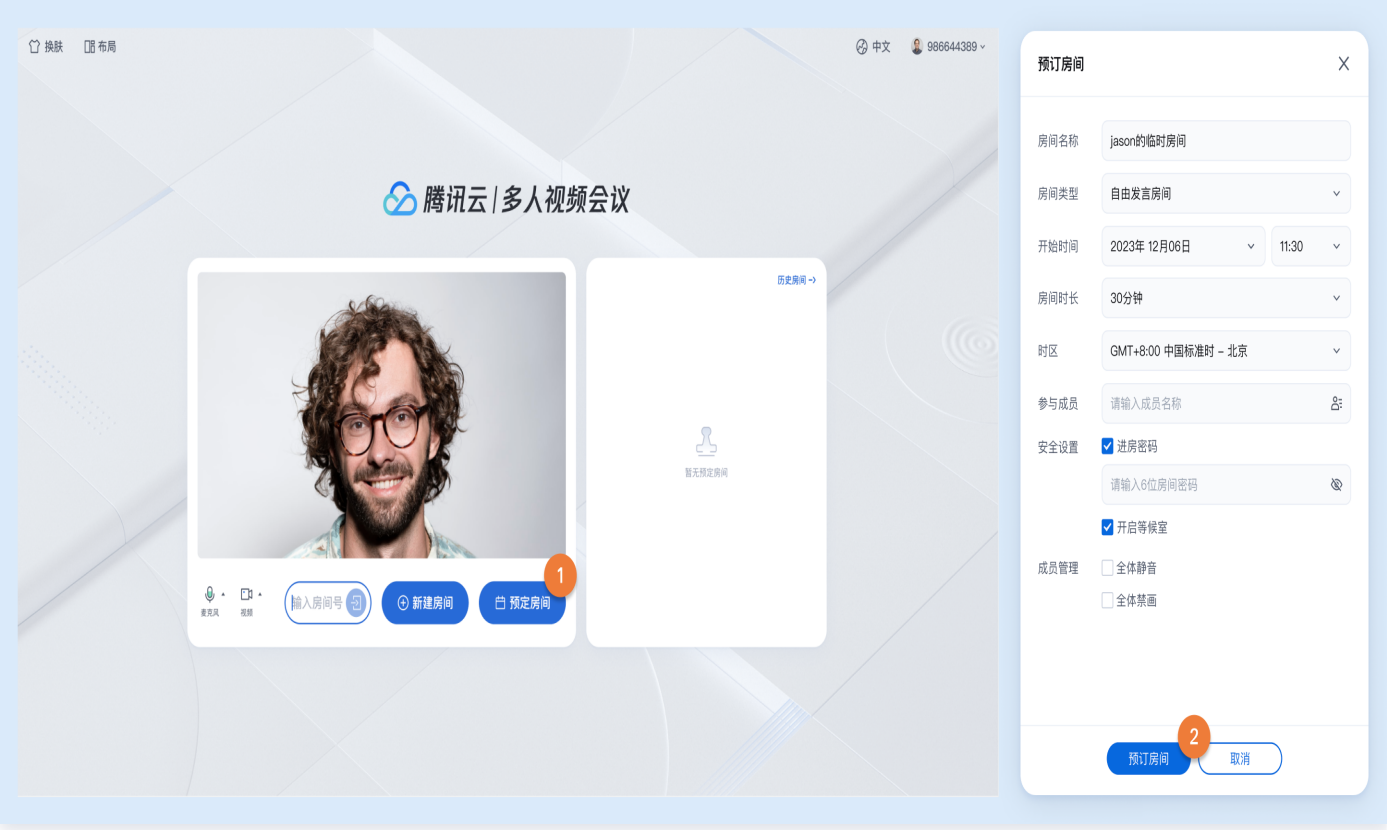
功能介绍

TUIRoomKit 推出了预定房间新特性，用户可预定一个房间，安排一场日程中的会议。

> **说明：**
> **TUIRoomKit 自 v2.5.0 版本支持预定房间、查看房间列表、修改房间信息等能力，集成最新版本 TUIRoomKit 可体验预定房间流程。**

如何预定房间

1. 在 TUIRoomKit 会前预览页面单击**预定房间** > 填写房间预定信息及设置相关权限 > 完成设置后单击**预定房间**即可。



2. 预定房间成功后将会同步预定结果呈现在会前预览页面右侧**预定房间列表**中，并支持**查看详情**、**修改房间**、**取消房间**、**复制邀请信息**等操作。另外，您还可以单击**加入**进入已预定的房间。

准备条件

在使用腾讯云提供的预定房间特性前，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 。接着，您需要引入 TUIRoomKit 组件，具体可参见 快速跑通 。

预定房间示例跑通

> **说明：**
> 您需要引入 PreConferenceView（**会前预览组件**）和 ConferenceMainView（**TUIRoomkit UI 组件主体**），示例中采用 v-if 和 v-else 指令来控制两个组件的展示与隐藏，您也可以采用路由跳转的方式进行组件之间的切换。
> - 在 PreConferenceView 组件中，通过设置 enable-scheduled-conference 属性的值，控制预定房间特性的展示或隐藏。另外，组件监听了 on-enter-room 事件，当用户点击进入房间时，触发 handleEnterRoom 方法即可调用 join 接口。

- 在 ConferenceMainView 组件中，监听了 on-destroy-room 事件，当房间被销毁时，触发 onDestroyRoom 方法。

Web

```
<template>
  <PreConferenceView
    v-if="isShowPreConferenceView"
    :enable-scheduled-conference="true" // 设置是否开启预定房间特性展示
    @on-enter-room="handleEnterRoom"
  ></PreConferenceView>
  <ConferenceMainView
    v-else
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  ></ConferenceMainView>
</template>

<script setup lang="ts">
import { ref } from 'vue';
 // 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 { PreConferenceView, conference, ConferenceMainView } from
'@tencentcloud/roomkit-web-vue3';

const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，
https://cloud.tencent.com/document/product/647/104842
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
    userId: '',
    // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool
页面快速生成 userSig，注意 userSig 与 userId 为一一对应关系
    userSig: '',
  });
}
init();

async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
```

```
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```

Electron

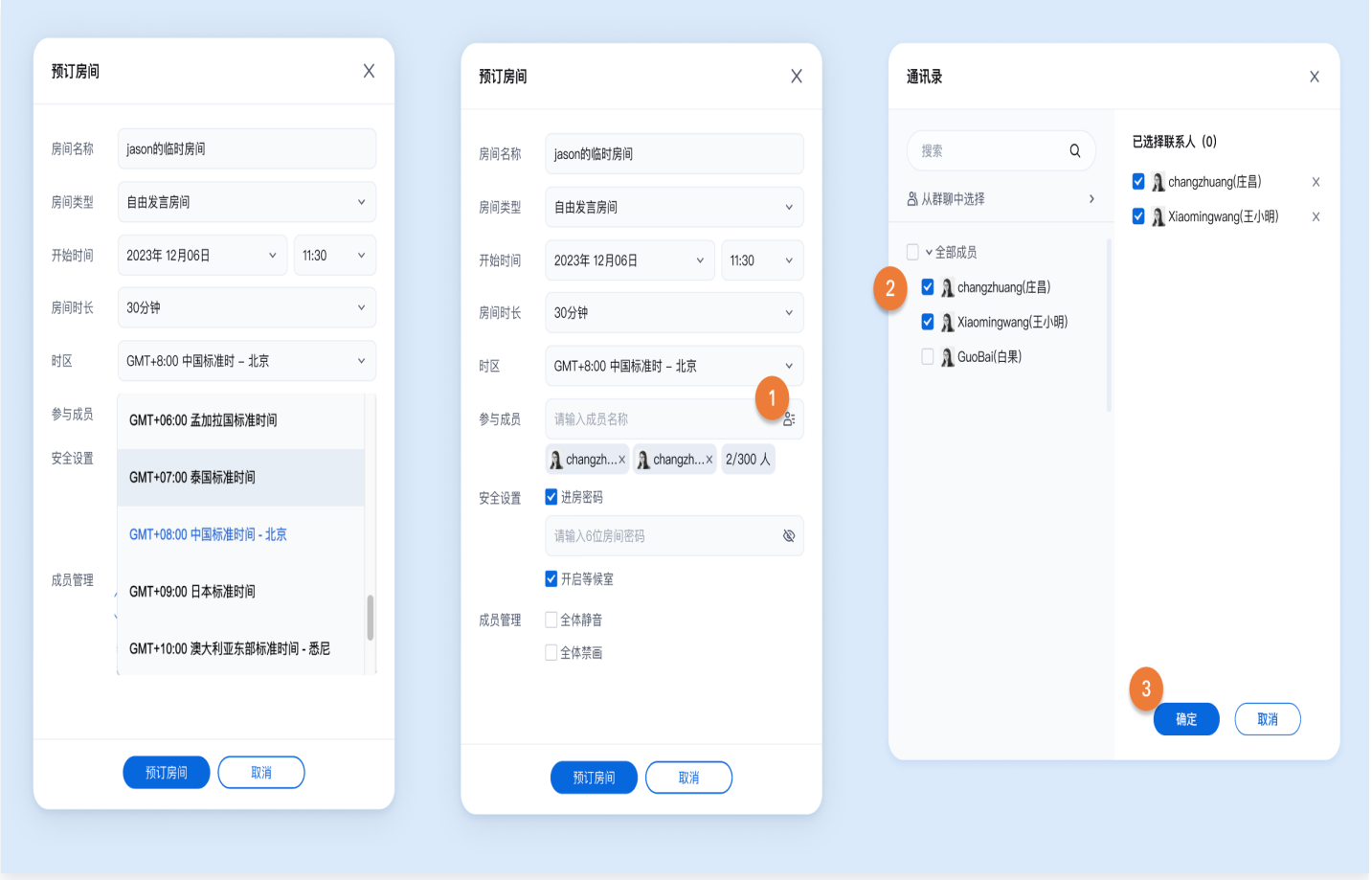
```
<template>
  <PreConferenceView
    v-if="isShowPreConferenceView"
    :enable-scheduled-conference="true" // 设置是否开启预定房间特性展示
    @on-enter-room="handleEnterRoom"
  ></PreConferenceView>
  <ConferenceMainView
    v-else
    display-mode="permanent"
    @on-destroy-room="onDestroyRoom"
  ></ConferenceMainView>
</template>

<script setup lang="ts">
import { ref } from 'vue';
 // 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 { PreConferenceView, conference, ConferenceMainView } from
'@tencentcloud/roomkit-electron-vue3';

const isShowPreConferenceView = ref(true);
const init = async () => {
  conference.login({
    // 获取 sdkAppId 可参考文档开通服务部分，
https://cloud.tencent.com/document/product/647/104842
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
```

```
    userId: '',
    // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool
页面快速生成 userSig, 注意 userSig 与 userId 为一一对应关系
    userSig: '',
  });
}
init();

async function handleEnterRoom(roomOption: Record<string, any>) {
  const { roomId } = roomOption;
  await conference.join(roomId, {
    isOpenCamera: false,
    isOpenMicrophone: false,
  });
  isShowPreConferenceView.value = false;
}

function onDestroyRoom() {
  isShowPreConferenceView.value = true;
  init();
}
</script>
```

预定房间控制

预定房间

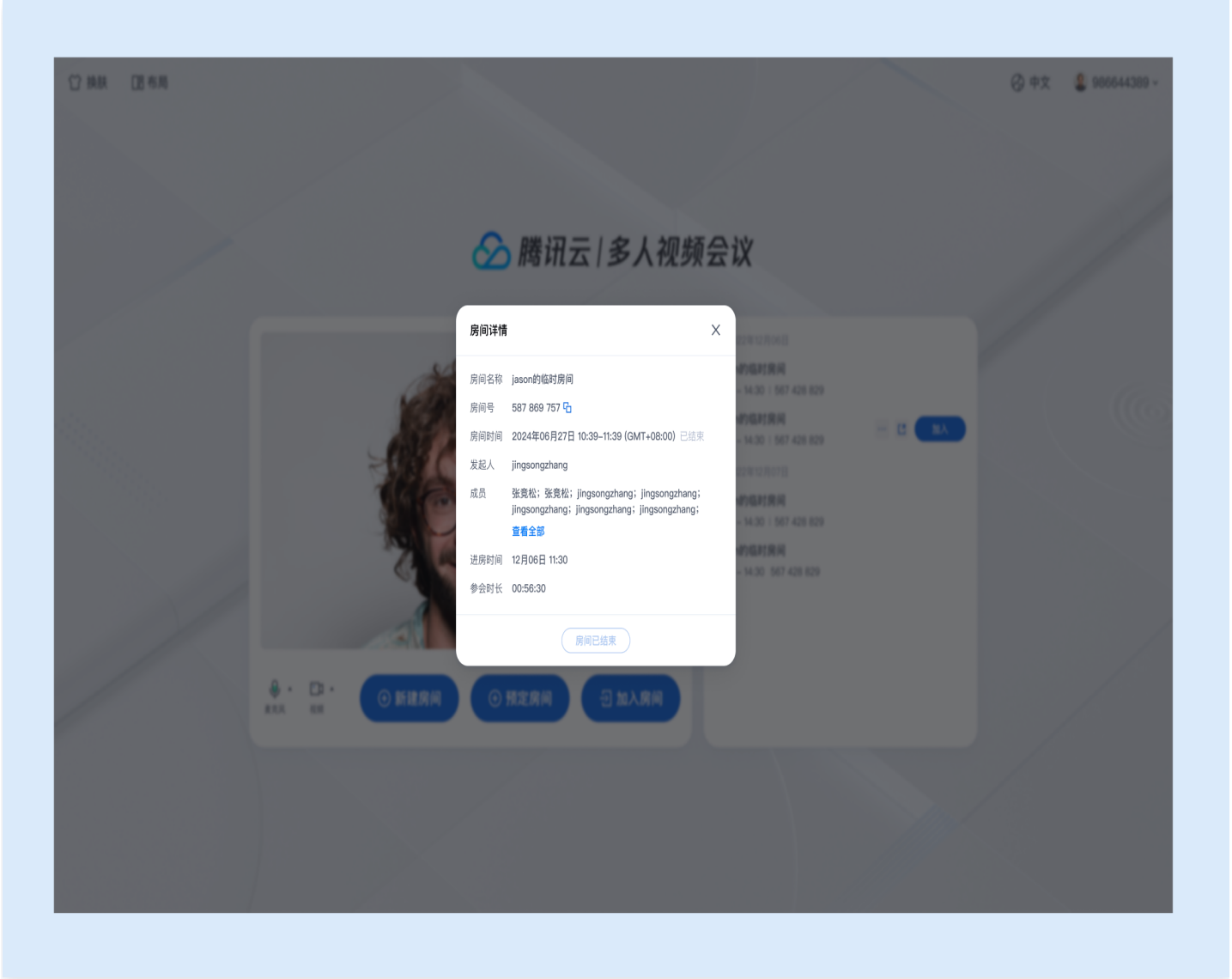
会前预览页单击**预定房间**后，弹出预定房间设置框，用户可根据自己的需求设置房间信息，其中可设置内容包含：**房间名称、房间类型、开始时间、房间时长、时区、参与成员、安全设置（进房密码）、成员管理（全体静音、全体禁画）**等。



> **注意：**
> TUIRoomKit 的邀请成员列表的成员信息来自于 IM **好友关系**，需要使用 IM **完成好友关系的添加**。您可以通过**添加 IM 好友关系**的方式来完成用户数据的替换。此时您需要通过 IM REST API 采用添加好友关系的方式得到 IM 好友关系链的数据。若您进行参会成员通讯录的数据导入或添加用户，可参见 添加好友 。若您需要移除参会成员，即删除 IM 关系链中的好友关系即可，具体可参见 删除好友 。

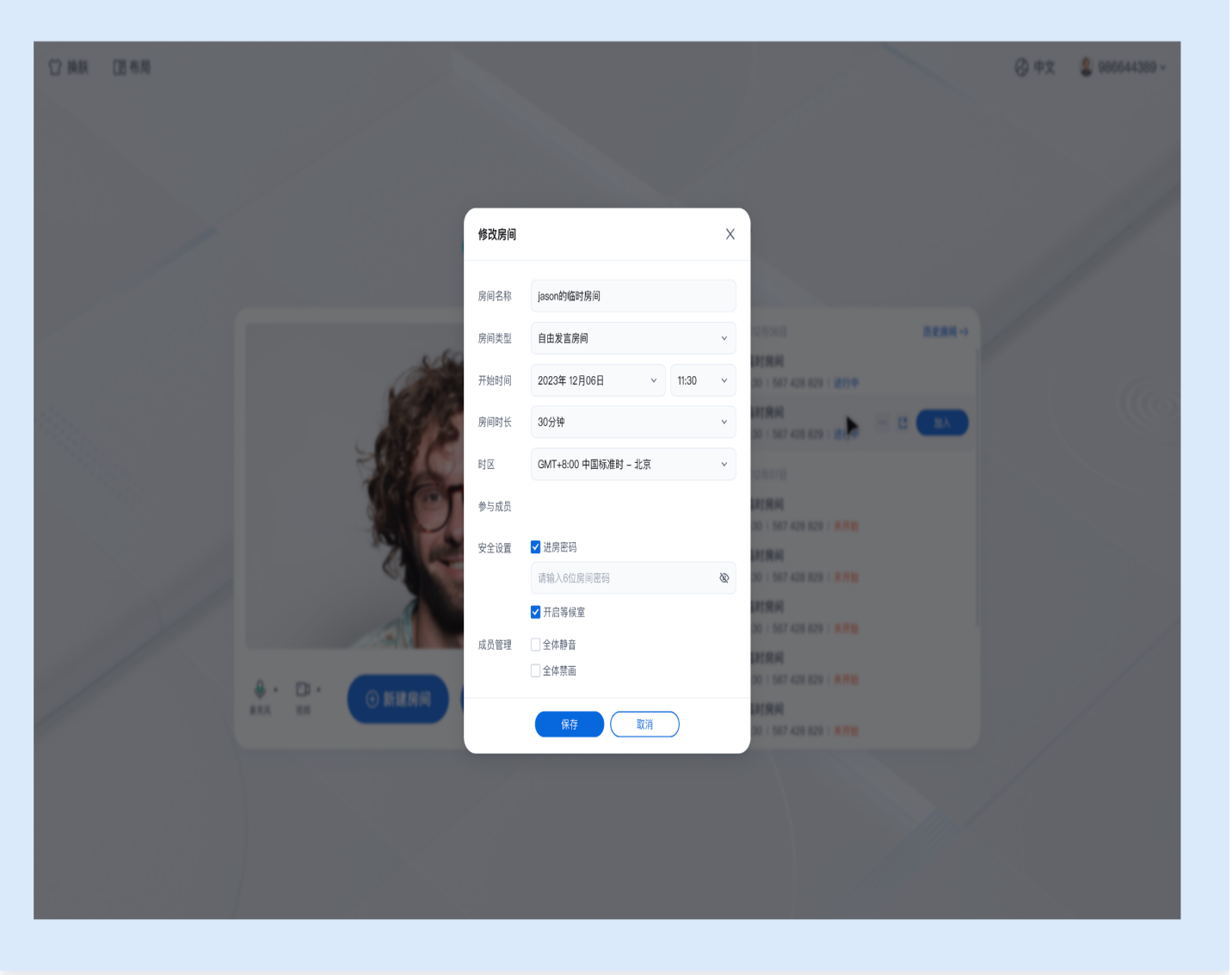
查看详情

用户可单击**查看详情**，查看对应预定房间的详细信息。



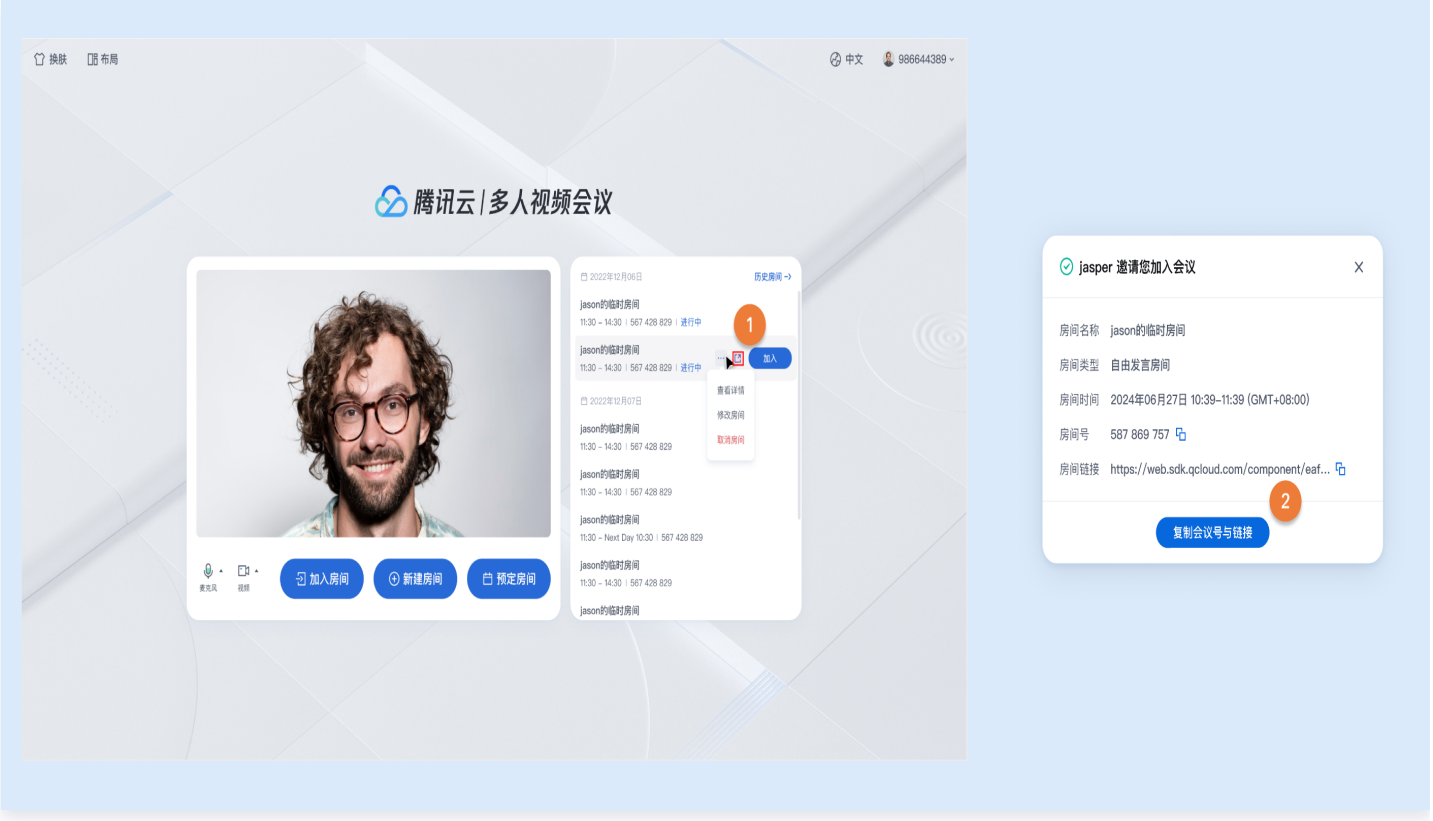
修改房间信息

房间拥有者可以对已预定的房间信息进行修改，修改完毕单击**保存**即可将当前预定房间的信息调整为修改后的房间信息。



邀请进入房间

用户可通过点击邀请弹出会议信息邀请框，通过单击**复制会议号与链接**，将预定房间信息复制到粘贴板分享给其他用户。



注意事项

- 预定房间开始时间不能早于当前时间，但没有提前天数的限制。
- 若同时预定不同日期/时间的房间，分次选择时间提交即可。
- 房间预定成功后，房间号将从预定开始时间起，若房间没有用户，那么房间保留6小时，在此期间您都可以随时回到该房间。
- 预定房间成功后，即可获取房间号和预定信息。

交流与反馈

如果您在接入和使用过程有任何需要或者反馈，欢迎加入我们的 TUIRoomKit 技术交流平台 zhiliao，进行技术交流和问题反馈。