多人音视频房间 SDK

会中呼叫

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

会中呼叫
Android&iOS

最近更新时间：2024-09-26 11:07:11

功能介绍

TUIRoomKit 支持会中呼叫功能。用户在会议进行过程中，可以随时呼叫其他用户加入当前会议，无需提前预定或安排。通过会中呼叫功能，用户可以灵活地邀请或提醒相关人员参与会议，提升会议的互动性和效率。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。

呼叫端	被呼叫端

使用说明

呼叫用户

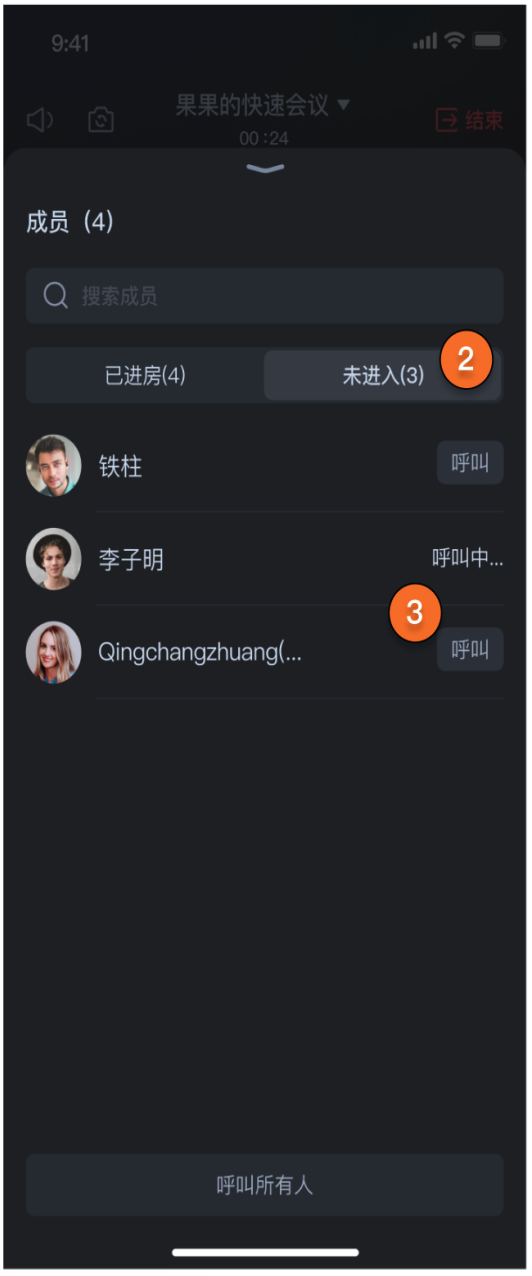
当您在会议中时，您可以通过如下两种方式对未进房的用户进行呼叫：

方式一：呼叫成员列表中的未进入用户

在房间的成员列表中，您会看到一个名为**未进入**的标题栏。点击**未进入**，会显示所有当前未进入会议的成员，您可以对这些尚未进入会议的成员进行呼叫。
未进入的列表中包含两类用户：

- 当前会议在预定时邀请且未进入的成员

- 已对其进行呼叫但仍未入会的成员



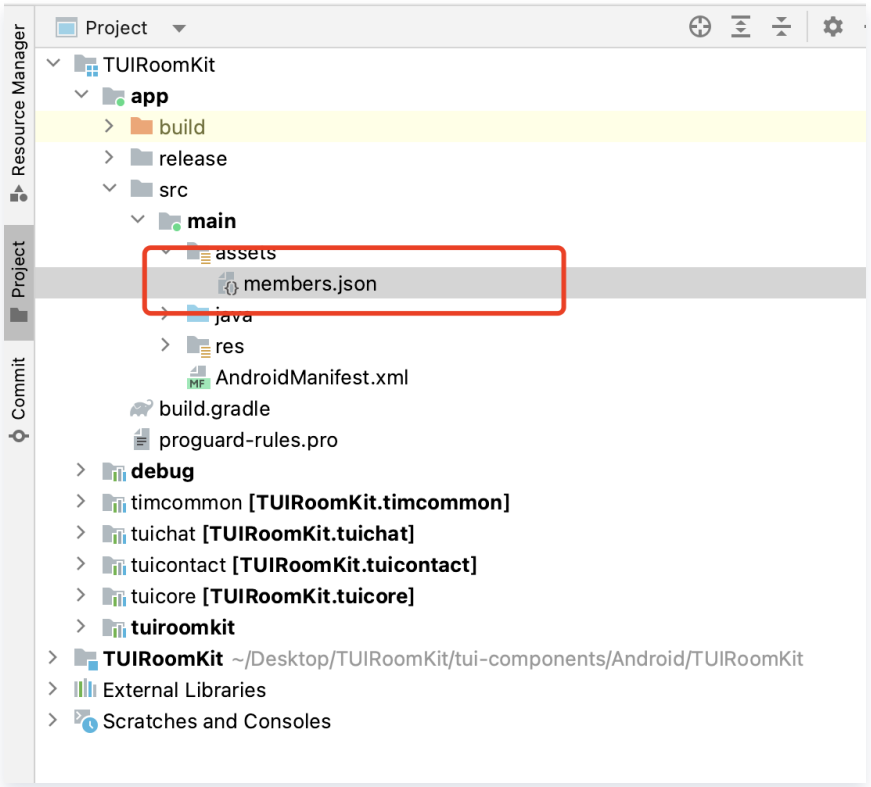
方式二：呼叫通讯录中的用户

通过点击底部栏中的**邀请** > **添加成员**，您可以唤起您自己的通讯录界面，并对其中您所选定的成员进行呼叫。
如您需要使用此功能，您需要通过如下方式，根据您的业务需求导入您自行实现的通讯录界面：

Android

如何体验呼叫通讯录成员的功能

首先，请参考 跑通 Demo 完成 Demo 的运行。在 Demo 项目的 `members.json` 文件中，我们已经预配置了一些测试用的用户信息。您可以选择两个账号，分别在两台手机上使用我们配置的 userId 登录，然后在会议中点击底部栏的**邀请 > 添加成员**以唤起通讯录，在通讯录中选择另一个用户并点击确认进行呼叫。这样，另一个用户就会收到您的呼叫。



如何使用自定义通讯录

1. **TUIRoomKit 关联自定义通讯录：**您需要在呼叫通讯录中的用户之前，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

```
Intent intent = new Intent();
// participants 为选择完毕的用户列表，必须为 ArrayList<User> 类型。
ConferenceParticipants participants = new ConferenceParticipants();
// 添加您的成员
...

intent.putExtra(SELECTED_PARTICIPANTS, participants);
setResult(3, intent);
finish();
```

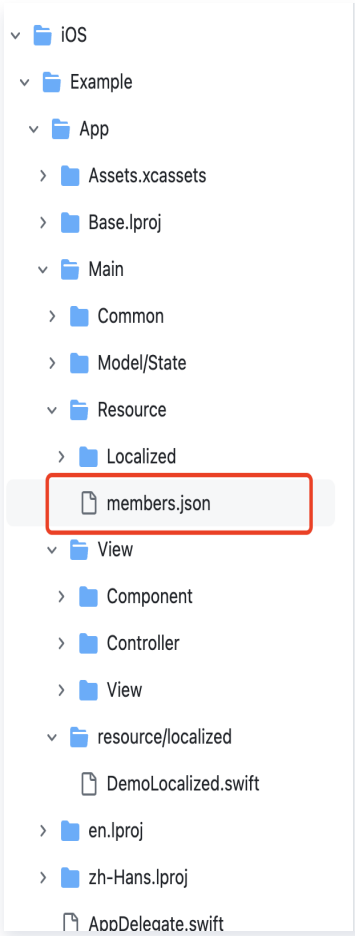
kotlin

```
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()
```

iOS

如何体验呼叫通讯录成员的功能

首先，请参考 跑通 Demo 完成 Demo 的运行。在 Demo 项目的 `members.json` 文件中，我们已经预配置了一些测试用的用户信息。您可以选择两个账号，分别在两台手机上使用我们配置的userId登录，然后在会议中点击底部栏的**邀请 > 添加成员**以唤起通讯录，在通讯录中选择另一个用户并点击确认进行呼叫。这样，另一个用户就会收到您的呼叫。




```
[
    {
        "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"
    }
]
```

如何使用自定义通讯录

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。

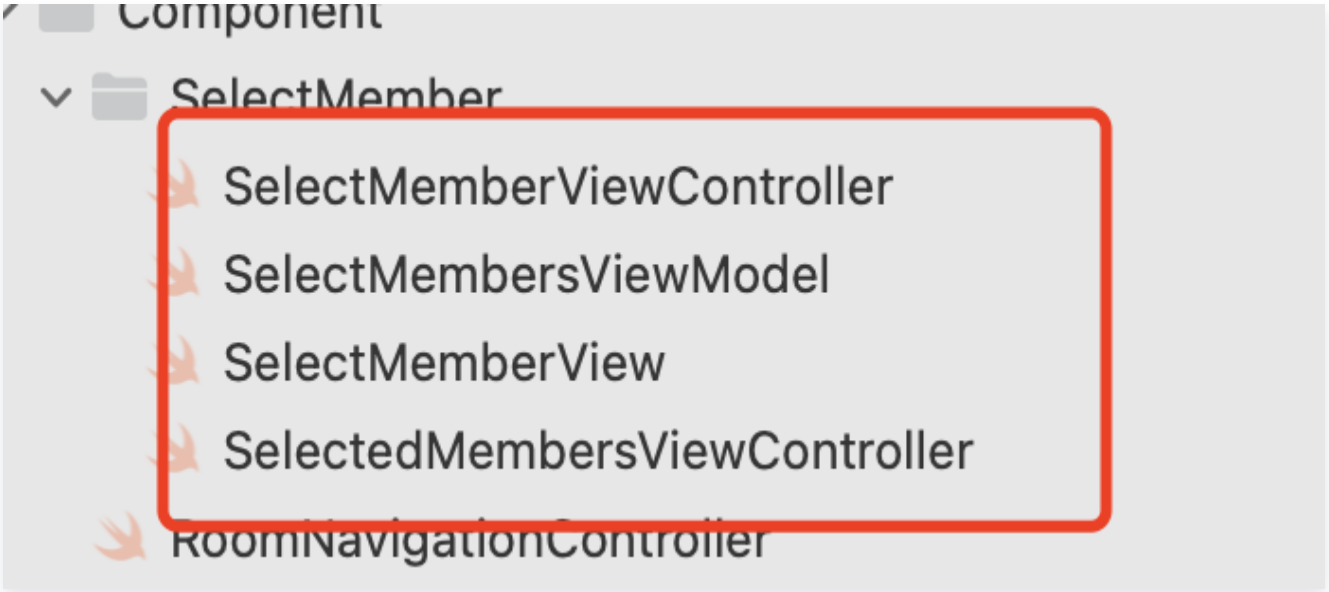
```
// 示例代码
class SelectMemberViewController: UIViewController,
ContactViewProtocol {
    weak var delegate: ContactViewSelectDelegate?
    var selectedList: [User]

    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：**
> 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。在会中呼叫时，不可选择的成员为已在会中的成员。

2. 在呼叫通讯录中的用户**之前**，您需要通过以下方法将您自定义的通讯录传入TUIRoomKit：

```
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通讯录页面了，同时我们在 demo 中提供了上方图片示例中通讯录的页面代码，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）。

收到呼叫

当您在应用内收到呼叫时，会弹出如下图所示的页面。您可以拖动滑块选择**立即加入**，或点击**暂不进入**以拒绝此呼叫。

> **说明：**
> 当用户**已在会议中**或**正在被呼叫**时，该用户会自动拒绝所有呼叫。

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，这里对会中呼叫功能相关的文件做了介绍。

自定义被呼叫页面视图

如您需要自定义被呼叫页面的视图，请参考以下路径进行更改：

Android

```
// 文件位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/component/

component
    └──InvitationReceivedView.java
```

iOS

```
// 文件位置：
iOS/TUIRoomKit/Source/View/ConferenceOptions/ConferenceInvitation

ConferenceInvitation
    └── ConferenceInvitationViewController.swift  // 被呼叫页面视图
```

自定义成员列表中呼叫视图

如您需要自定义成员列表中呼叫成员的视图，请参考以下路径进行更改：

Android

```
// 文件位置：
Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit/roomkit/view/page/widget/UserControlPanel/

UserControlPanel
    └── CallUserView.java // 成员列表呼叫按钮
```

iOS

```
// 文件位置：iOS/TUIRoomKit/Source/Page/Widget/UserControlPanel

UserControlPanel              // 成员列表相关的视图目录
    └── UserListCell.swift    // 成员列表中单个成员视图，包含用户呼叫状态视图
```

关键代码

呼叫用户

Android

```
// 文件位置：
TUIRoomKit/blob/main/Android/tuiroomkit/src/main/java/com/tencent/cl
oud/tuikit/roomkit/model/controller/InvitationController.java

public void inviteUsers(List<UserState.UserInfo> userInfoList,
TUIConferenceInvitationManager.InviteUsersCallback callback) {
    Log.d(TAG, "inviteUsers");
    if (userInfoList.isEmpty()) {
        return;
    }

RoomToast.toastShortMessageCenter(TUILogin.getAppContext().getString
(R.string.tuiroomkit_invitation_has_been_sent));

mConferenceInvitationManager.inviteUsers(mRoomState.roomId.get(),
getUserIdListFromUserList(userInfoList), INVITE_TIME_OUT_SECONDS,
"", new TUIConferenceInvitationManager.InviteUsersCallback() {
        @Override
        public void onSuccess(Map<String,
TUIConferenceInvitationManager.InvitationCode> invitationResultMap)
{
            Log.d(TAG, "inviteUsers success");
            if (callback != null) {
                callback.onSuccess(invitationResultMap);
```

```
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String
message) {
            Log.d(TAG, "inviteUsers error=" + error + " message=" +
message);
            if (callback != null) {
                callback.onError(error, message);
            }
        }
    });
}
```

iOS

```
// 文件位置:
TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService
.swift

func inviteUsers(roomId: String, userIdList: [String]) ->
AnyPublisher<InviteUsersResult, RoomError> {
    return Future<InviteUsersResult, RoomError> { [weak self]
promise in
        guard let self = self else { return }
        self.invitationManager?.inviteUsers(roomId, userIdList:
userIdList, timeout: timeout, extensionInfo: "") {dic in
            promise(.success((dic)))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message:
message)))
        }
    }
    .eraseToAnyPublisher()
}
```

接受呼叫

Android

```
// 文件位置：
TUIRoomKit/blob/main/Android/tuiroomkit/src/main/java/com/tencent/cl
oud/tuikit/roomkit/model/controller/InvitationController.java

public void accept(String roomId, TUIRoomDefine.ActionCallback
callback) {
    Log.d(TAG, "accept");
    mConferenceInvitationManager.accept(roomId, new
TUIRoomDefine.ActionCallback() {
        @Override
        public void onSuccess() {
            Log.d(TAG, "accept success");
            if (callback != null) {
                callback.onSuccess();
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String
message) {
            Log.d(TAG, "accept error=" + error + " message=" +
message);
            if (callback != null) {
                callback.onError(error, message);
            }
        }
    });
}
```

iOS

```
// 文件位置：
TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService
.swift

func accept(roomId: String) -> AnyPublisher<String, RoomError> {
```

```
    return Future<String, RoomError> { [weak self] promise in
        guard let self = self else { return }
        self.invitationManager?.accept(roomId) {
            promise(.success(roomId))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message:
message)))
        }
    }
    .eraseToAnyPublisher()
}
```

拒绝呼叫

Android

```
// 文件位置:
TUIRoomKit/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit
/roomkit/model/controller/InvitationController.java

public void reject(String roomId,
TUIConferenceInvitationManager.RejectedReason reason,
TUIRoomDefine.ActionCallback callback) {
    Log.d(TAG, "reject roomId= " + roomId + " reason=" + reason);
    mConferenceInvitationManager.reject(roomId, reason, new
TUIRoomDefine.ActionCallback() {
        @Override
        public void onSuccess() {
            Log.d(TAG, "reject success");
            if (callback != null) {
                callback.onSuccess();
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String
message) {
            Log.d(TAG, "reject error=" + error + " message=" +
message);
            if (callback != null) {
```

```
                callback.onError(error, message);
            }
        }
    });
}
```

iOS

```
// 文件位置:
TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService
.swift

func reject(roomId: String, reason: TUIInvitationRejectedReason) ->
AnyPublisher<String, RoomError> {
    return Future<String, RoomError> { [weak self] promise in
        guard let self = self else { return }
        self.invitationManager?.reject(roomId, reason: reason) {
            promise(.success(roomId))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message:
message)))
        }
    }
    .eraseToAnyPublisher()
}
```

获取房间内呼叫列表

Android

```
// 文件位置:
TUIRoomKit/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit
/roomkit/model/controller/InvitationController.java

private void getInvitationList() {
    Log.d(TAG, "getInvitationList");
```

```
mConferenceInvitationManager.getInvitationList(mRoomState.roomId.get
(), getAttendeeListCursor, SINGLE_FETCH_COUNT, new
TUIConferenceInvitationManager.GetInvitationListCallback() {
        @Override
        public void
onSuccess(TUIConferenceInvitationManager.InvitationListResult
invitationListResult) {
            Log.d(TAG, "getInvitationList");
            for (TUIConferenceInvitationManager.Invitation
invitation : invitationListResult.invitationList) {
                InvitationState.Invitation invitationState = new
InvitationState.Invitation();
                invitationState.invitee = new
UserState.UserInfo(invitation.invitee);
                invitationState.inviter = new
UserState.UserInfo(invitation.inviter);
                invitationState.invitationStatus =
invitation.status;

mInvitationState.invitationList.add(invitationState);
            }
            getInvitationListCursor = invitationListResult.cursor;
            if (!"".equals(getInvitationListCursor)) {
                getInvitationList();
            }
        }

        @Override
        public void onError(TUICommonDefine.Error error, String
message) {
            Log.d(TAG, "getInvitationList onError error=" + error +
" message=" + message);
        }
    });
}
```

iOS

```
// 文件位置：
TUIRoomKit/iOS/TUIRoomKit/Source/Service/ConferenceInvitationService
.swift

func getInvitationList(roomId: String, cursor: String, count: Int =
20) -> AnyPublisher<InvitationfetchResult, RoomError> {
    return Future<InvitationfetchResult, RoomError> { [weak self]
promise in
        guard let self = self else { return }
        self.invitationManager?.getInvitationList(roomId, cursor:
cursor, count: count) {invitations, cursor in
            promise(.success((invitations, cursor)))
        } onError: { error, message in
            promise(.failure(RoomError(error: error, message:
message)))
        }
    }
    .eraseToAnyPublisher()
}
```

用户收到呼叫监听

Android

```
// 文件位置：
TUIRoomKit/Android/tuiroomkit/src/main/java/com/tencent/cloud/tuikit
/roomkit/model/ConferenceServiceInitializer.java

private void initConferenceInvitationObserver() {
    TUIConferenceInvitationManager invitationManager =
(TUIConferenceInvitationManager)
TUIRoomEngine.sharedInstance().getExtension(TUICommonDefine.Extensio
nType.CONFERENCE_INVITATION_MANAGER);
    invitationManager.addObserver(new
TUIConferenceInvitationManager.Observer() {
        @Override
        public void onReceiveInvitation(TUIRoomDefine.RoomInfo
roomInfo, TUIConferenceInvitationManager.Invitation invitation,
```

```
String extensionInfo) {
            if
(ConferenceController.sharedInstance().getViewState().isInvitationPe
nding.get()) {

ConferenceController.sharedInstance().getInvitationController().reje
ct(roomInfo.roomId, REJECT_TO_ENTER, null);
                return;
            }
            if
(ConferenceController.sharedInstance().getRoomController().isInRoom(
)) {

ConferenceController.sharedInstance().getInvitationController().reje
ct(roomInfo.roomId, IN_OTHER_CONFERENCE, null);
                return;
            }

            Bundle bundle = new Bundle();
            bundle.putString("roomId", roomInfo.roomId);
            bundle.putString("conferenceName", roomInfo.name);
            bundle.putString("ownerName", roomInfo.ownerName);
            bundle.putString("inviterName",
invitation.inviter.userName);
            bundle.putString("inviterAvatarUrl",
roomInfo.ownerAvatarUrl);
            bundle.putInt("memberCount", roomInfo.memberCount);
            TUICore.startActivity("InvitationReceivedActivity",
bundle);
        }
    });
}
```

iOS

```
// 文件位置:
TUIRoomKit/iOS/TUIRoomKit/Source/Service/InvitationObserverService.s
wift
```

```
func onReceiveInvitation(roomInfo: TUIRoomInfo, invitation:
TUIInvitation, extensionInfo: String) {
    let store = Container.shared.conferenceStore()
    store.dispatch(action:
ConferenceInvitationActions.onReceiveInvitation(payload: (roomInfo,
invitation)))
}
```