

# 多人音视频房间 SDK

## 屏幕共享



腾讯云

## 【 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 】



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

屏幕共享

Android&iOS&Flutter

# 屏幕共享

## Android&iOS&Flutter

最近更新时间：2025-06-18 11:14:52

本文档将详细介绍 TUIRoomKit 屏幕共享功能，帮助您更好地掌握 TUIRoomKit 会议过程中的屏幕共享相关功能操作。通过本文档，您能够充分利用 TUIRoomKit 的功能，实现高质量的音视频会议。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屏幕共享相关的交互功能，可以接入 TUIRoomEngineSDK，并参见 [关键代码](#) 相关调用来实现您的需求。

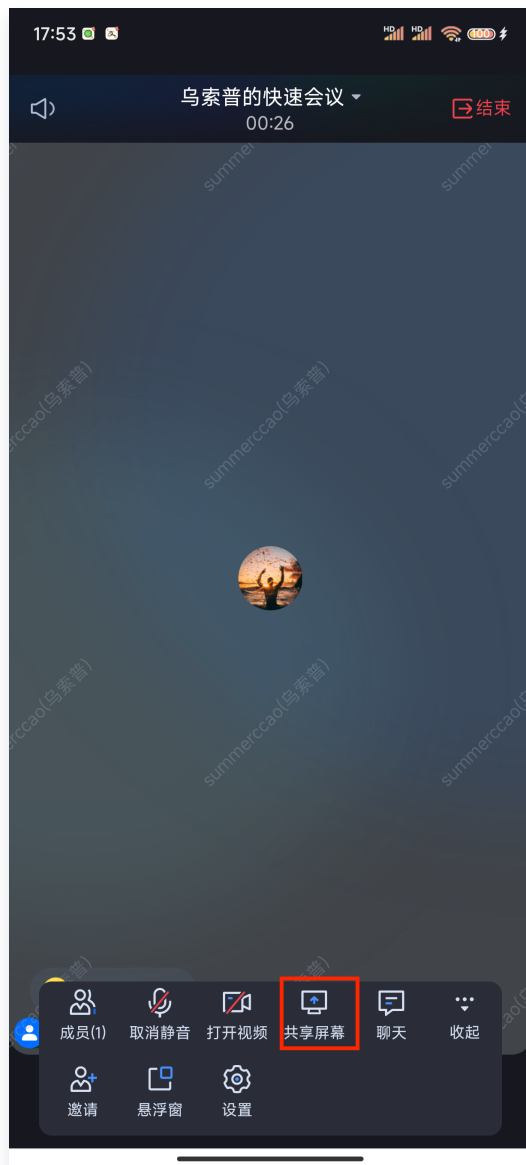
### 功能介绍

TUIRoomKit 支持屏幕共享，用户可以在进入房间之后分享自己的屏幕，房间内的其他成员可以观看正在分享中的屏幕。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。

### 使用说明

Android、iOS 和 Flutter 端的用户可以通过点击底部工具栏的**共享屏幕**按钮来实现屏幕共享操作。

屏幕



## 功能接入

在接入屏幕共享功能时，您可能需要进行一些配置。不同平台的配置方法请参见：

### Android

由于 Android 系统隐私策略的更改，在 Android 10 及以上版本 App 若使用录屏等功能需要在前台 Service 中进行，否则录屏/屏幕共享时系统将报错或 App 被系统终止。在我们的组件中，会自动为您开启该 **Android Foreground Service**，您只需要点击**分享屏幕按钮**，同意相关隐私政策后即可进行共享。

### iOS

iOS 系统上的跨应用屏幕分享，需要增加 Extension 录屏进程以配合主 App 进程进行推流。Extension 录屏进程由系统在需要录屏的时候创建，并负责接收系统采集到屏幕图像。因此需要：

1. 创建 App Group，并在 Xcode 中进行配置（可选）。这一步的目的是让 Extension 录屏进程可以与主 App 进程进行跨进程通信。
2. 在您的工程中，新建一个 Broadcast Upload Extension 的 Target，并在其中集成 SDK 压缩包中专门为扩展模块定制的 `TXLiteAVSDK_ReplayKitExt.framework`。
3. 点击屏幕分享按钮，开始分享您的屏幕。

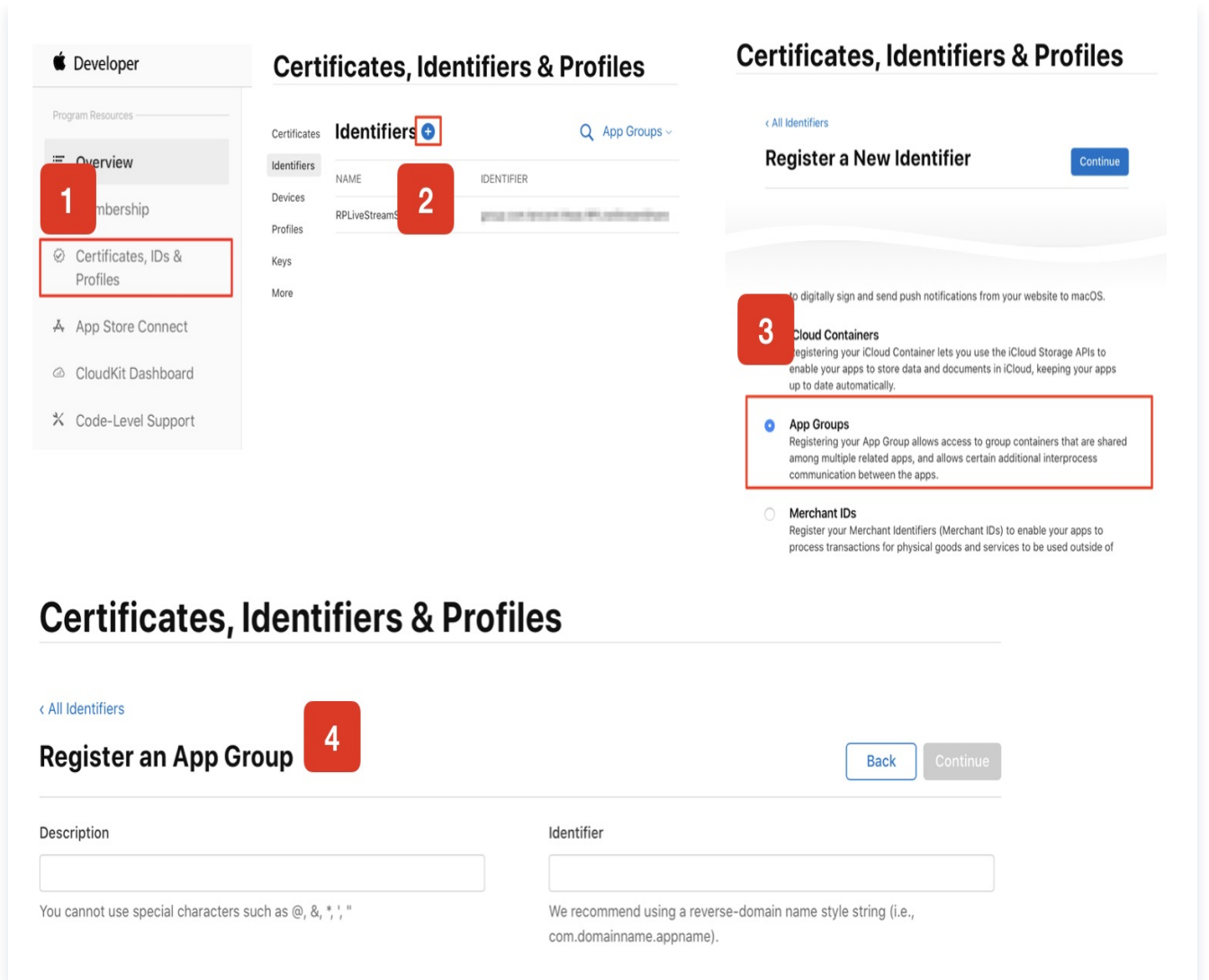
#### ⚠ 注意：

如果跳过步骤1，也就是不配置 App Group（接口传 null），屏幕分享依然可以运行，但稳定性要打折扣，故虽然步骤较多，但请尽量配置正确的 App Group 以保障屏幕分享功能的稳定性。

### 步骤1：创建 App Group

使用您的账号 [登录](#)，进行以下操作，注意完成后需要重新下载对应的 Provisioning Profile。

1. 单击 **Certificates, IDs & Profiles**。
2. 在右侧的界面中单击加号。
3. 选择 **App Groups**，单击 **Continue**。
4. 在弹出的表单中填写 **Description** 和 **Identifier**，其中 **Identifier** 需要传入接口中的对应的 **AppGroup** 参数。完成后单击 **Continue**。



5. 回到 **Identifier** 页面，左上边的菜单中选择 **App IDs**，然后单击您的 **App ID**（主 App 与 Extension 的 App ID 需要进行同样的配置）。
6. 选中 **App Groups** 并单击 **Edit**。
7. 在弹出的表单中选择您之前创建的 **App Group**，单击 **Continue** 返回编辑页，单击 **Save** 保存。

## Certificates, Identifiers & Profiles

## Certificates, Identifiers & Profiles

The screenshot displays the 'Identifiers' management interface. On the left, a table lists identifiers with columns for 'NAME' and 'IDENTIFIER'. The identifier 'liteavdemo' with ID 'com.tencent.liteavdemo' is highlighted with a red box and a red '5'. Below the table, the 'Capabilities' section is visible, showing a list of capabilities. The 'App Groups' capability is checked and highlighted with a red box and a red '6', with its 'Edit' button also highlighted.

## App Group Assignment

Select the App Groups you wish to assign to the bundle.

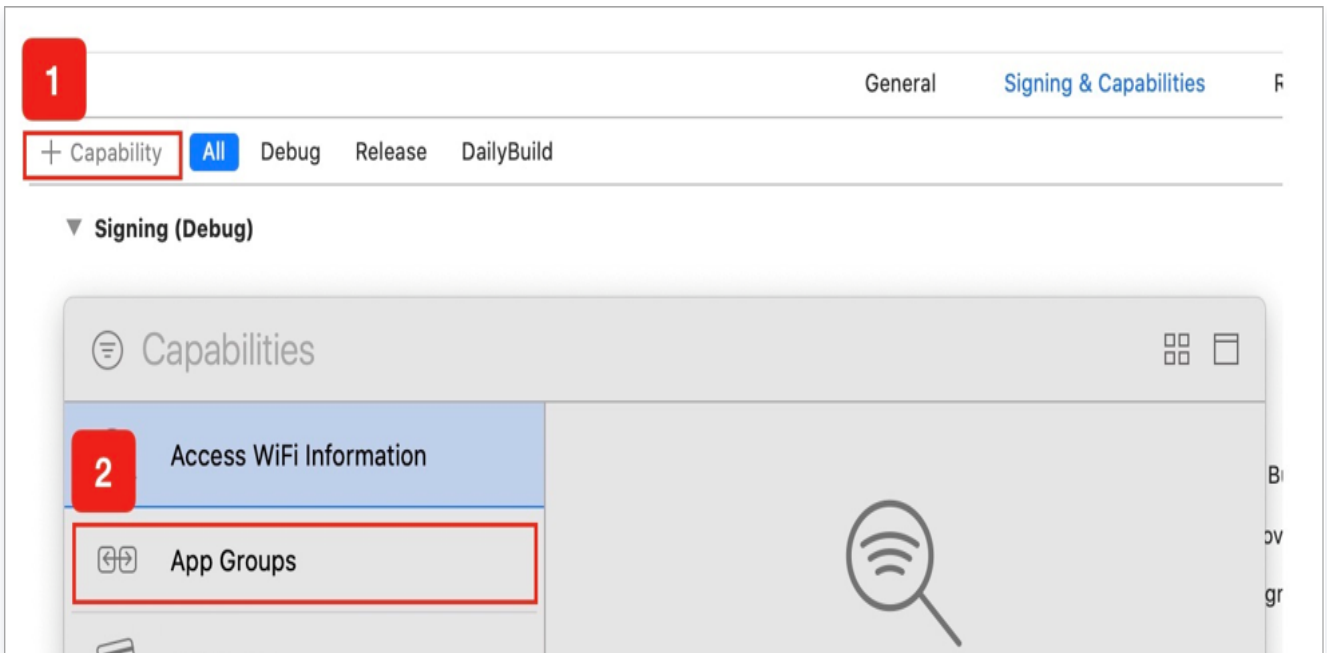
The screenshot shows the 'App Group Assignment' dialog. At the top, the 'Select All' checkbox is checked and highlighted with a red box and a red '7'. Below it, the 'RPLiveStreamShare' checkbox is also checked. The text '1 of 1 item(s) selected' is displayed on the right side of the dialog.

8. 重新下载 Provisioning Profile 并配置到 Xcode 中。

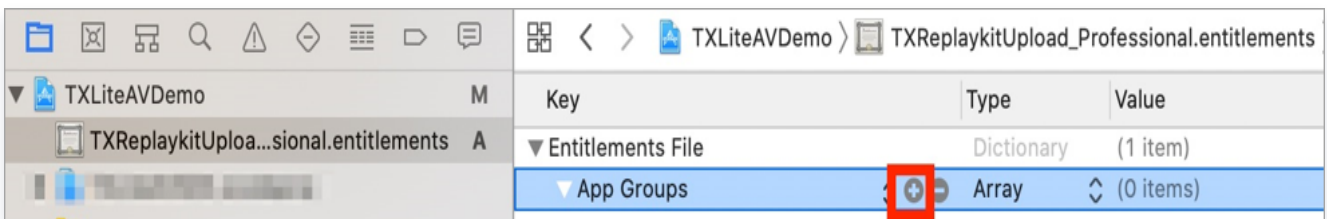
### 步骤2: 创建 Broadcast Upload Extension

1. 在 Xcode 菜单依次单击 **File > New > Target...**，选择 **Broadcast Upload Extension**。
2. 在弹出的对话框中填写相关信息，不用勾选 **Include UI Extension**，单击 **Finish** 完成创建。
3. 将下载到的 SDK 压缩包中的 **TXLiteAVSDK\_ReplayKitExt.framework** 拖动到工程中，勾选刚创建的 Target。
4. 选中新增加的 Target，依次单击 **+ Capability**，双击 **App Groups**，如下图：





操作完成后，会在文件列表中生成一个名为 Target.entitlements 的文件，如下图所示，选中该文件并单击 + 号填写上述步骤中的 App Group 即可。



5. 选中主 App 的 Target，并按照上述步骤对主 App 的 Target 做同样的处理。

6. 在新创建的 Target 中，Xcode 会自动创建一个名为 "SampleHandler.swift" 的文件，用如下代码进行替换。需将代码中的 APPGROUP 改为上文中的创建的 App Group Identifier。

```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt

let APPGROUP = "group.com.tencent.comm.trtc.demo"

class SampleHandler: RPBroadcastSampleHandler,
TXReplayKitExtDelegate {

    let recordScreenKey =
Notification.Name.init("TRTCRecordScreenKey")

    override func broadcastStarted(withSetupInfo setupInfo:
[String : NSObject]?) {
        // User has requested to start the broadcast. Setup info
        from the UI extension can be supplied but optional.
    }
}
```

```
        TXReplayKitExt.sharedInstance().setup(withAppGroup:
APPGROUP, delegate: self)
    }

    override func broadcastPaused() {
        // User has requested to pause the broadcast. Samples will
stop being delivered.
    }

    override func broadcastResumed() {
        // User has requested to resume the broadcast. Samples
delivery will resume.
    }

    override func broadcastFinished() {
        // User has requested to finish the broadcast.
        TXReplayKitExt.sharedInstance().finishBroadcast()
    }

    func broadcastFinished(_ broadcast: TXReplayKitExt, reason:
TXReplayKitExtReason) {
        var tip = ""
        switch reason {
        case TXReplayKitExtReason.requestedByMain:
            tip = "屏幕共享已结束"
            break
        case TXReplayKitExtReason.disconnected:
            tip = "应用断开"
            break
        case TXReplayKitExtReason.versionMismatch:
            tip = "集成错误 (SDK 版本号不符合)"
            break
        default:
            break
        }

        let error = NSError(domain:
NSStringFromClass(self.classForCoder), code: 0, userInfo:
[NSLocalizedStringFailureReasonErrorKey:tip])
        finishBroadcastWithError(error)
    }
}
```

```
override func processSampleBuffer(_ sampleBuffer:
CMSampleBuffer, with sampleBufferType: RPSampleBufferType) {
    switch sampleBufferType {
    case RPSampleBufferType.video:
        // Handle video sample buffer
        TXReplayKitExt.sharedInstance()
        .sendVideoSampleBuffer(sampleBuffer)
        break
    case RPSampleBufferType.audioApp:
        // Handle audio sample buffer for app audio
        break
    case RPSampleBufferType.audioMic:
        // Handle audio sample buffer for mic audio
        break
    @unknown default:
        // Handle other sample buffer types
        fatalError("Unknown type of sample buffer")
    }
}
}
```

## Flutter

**Android:** 由于 Android 系统隐私策略的更改，在 Android 10 及以上版本 App 若使用录屏等功能需要在前台 Service 中进行，否则录屏/屏幕共享时系统将报错或 App 被系统终止。在我们的组件中，会自动为您开启该 Android Foreground Service，您只需要点击屏幕共享按钮，同意相关隐私政策后即可进行共享。Android Foreground Service 相关代码可参见我们的 [代码示例](#)。

**iOS:** iOS 系统上的跨应用屏幕分享，需要增加 Extension 录屏进程以配合主 App 进程进行推流。

Extension 录屏进程由系统在需要录屏的时候创建，并负责接收系统采集到屏幕图像。因此需要：

1. 创建 App Group，并在 Xcode 中进行配置（可选）。这一步的目的是让 Extension 录屏进程可以与主 App 进程进行跨进程通信。
2. 在您的工程中，新建一个 Broadcast Upload Extension 的 Target，并在其中集成 SDK 压缩包中专门为扩展模块定制的 `TXLiteAVSDK_ReplayKitExt.framework`。
3. 在组件中使用您的 App Group 和 Broadcast Upload Extension name 替换我们应用的 App Group 和 Broadcast Upload Extension name，代码位置请点击 [文件链接](#)。
4. 点击屏幕分享按钮，开始分享您的屏幕。

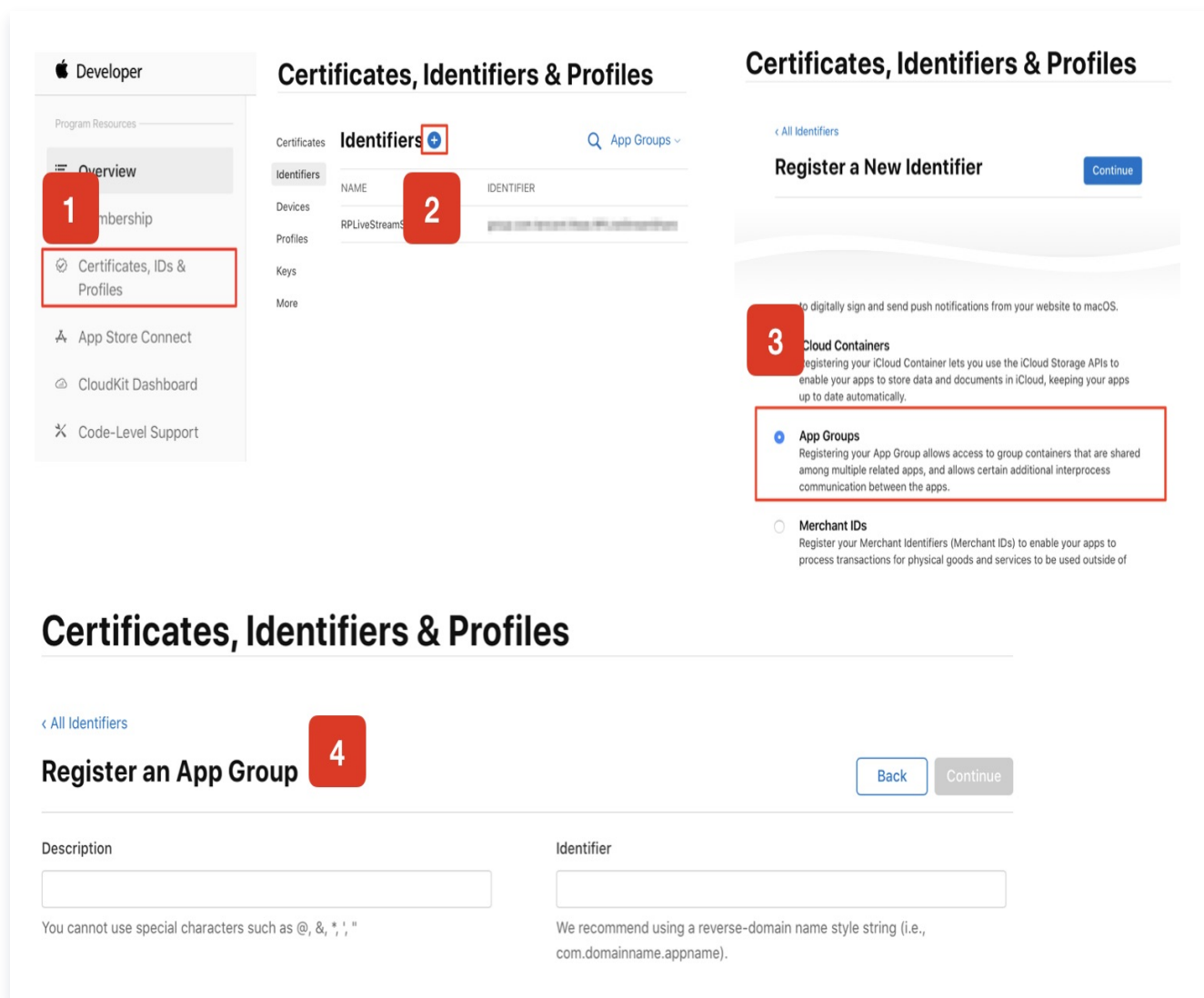
**注意：**

如果跳过步骤1，也就是不配置 App Group（接口传 null），屏幕分享依然可以运行，但稳定性要打折扣，故虽然步骤较多，但请尽量配置正确的 App Group 以保障屏幕分享功能的稳定性。

## 步骤1: 创建 App Group

使用您的账号 [登录](#)，进行以下操作，注意完成后需要重新下载对应的 Provisioning Profile。

1. 单击 **Certificates, IDs & Profiles**。
2. 在右侧的界面中单击加号。
3. 选择 **App Groups**，单击 **Continue**。
4. 在弹出的表单中填写 **Description** 和 **Identifier**，其中 **Identifier** 需要传入接口中的对应的 **AppGroup** 参数。完成后单击 **Continue**。



5. 回到 **Identifier** 页面，左上边的菜单中选择 **App IDs**，然后单击您的 App ID（主 App 与 Extension 的 App ID 需要进行同样的配置）。
6. 选中 **App Groups** 并单击 **Edit**。
7. 在弹出的表单中选择您之前创建的 **App Group**，单击 **Continue** 返回编辑页，单击 **Save** 保存。

## Certificates, Identifiers & Profiles

## Certificates, Identifiers & Profiles

Identifiers

NAME	IDENTIFIER
liteavdemo	com.tencent.liteavdemo
liteavdemoReplaykitUpload	com.tencent.liteavdemo.ReplaykitUpload

Platform: iOS, macOS, tvOS, watchOS

App ID Prefix: 5GHU44C.JHG (Team ID)

Description: liteavdemo

Bundle ID: com.tencent.liteavdemo (explicit)

Capabilities

ENABLED	NAME
<input type="checkbox"/>	Access WiFi Information
<input checked="" type="checkbox"/>	App Groups
<input type="checkbox"/>	Apple Pay Payment Processing

## App Group Assignment

Select the App Groups you wish to assign to the bundle.

Select All

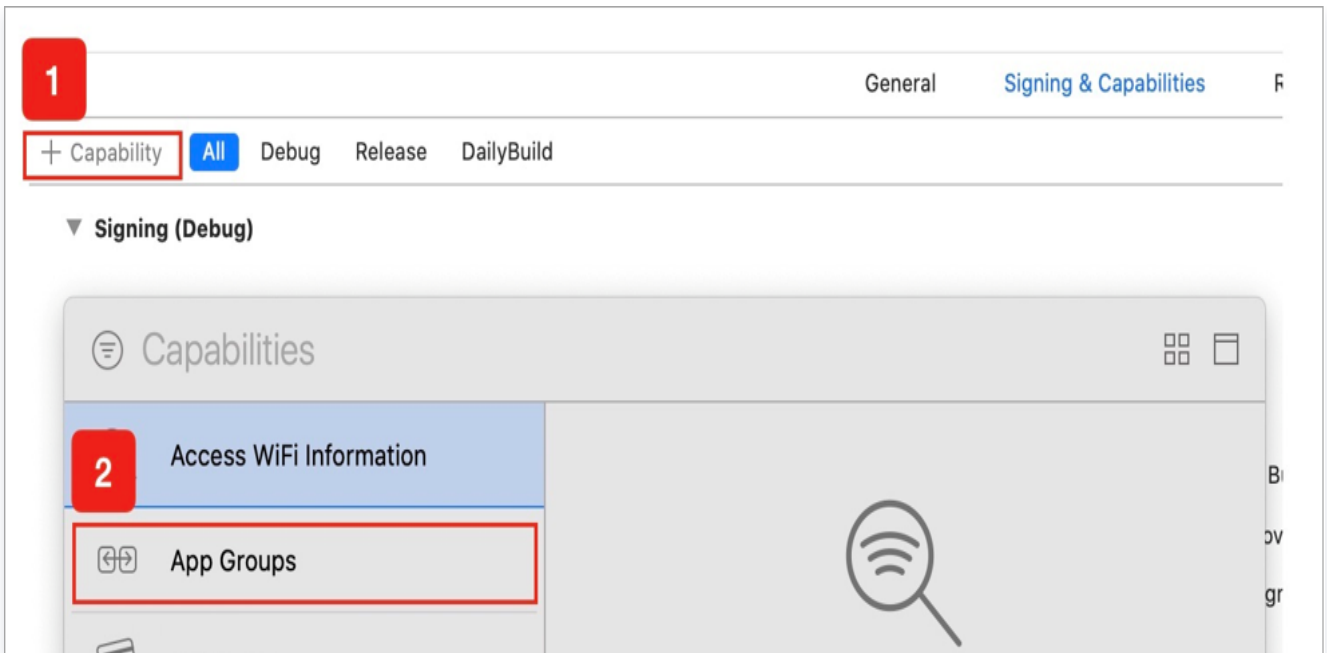
1 of 1 item(s) selected

RPLiveStreamShare

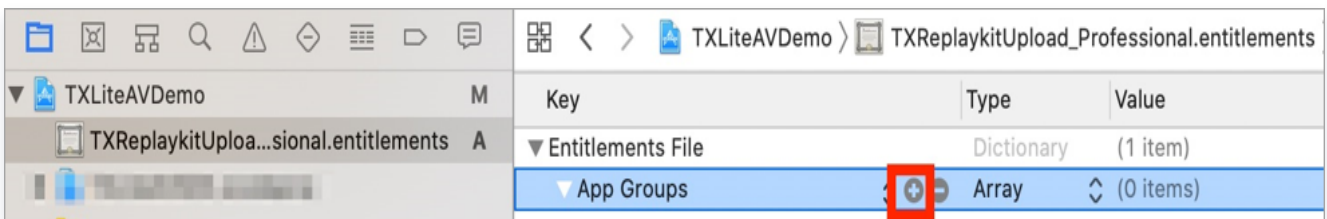
8. 重新下载 Provisioning Profile 并配置到 Xcode 中。

### 步骤2: 创建 Broadcast Upload Extension

1. 在 Xcode 菜单依次单击 **File > New > Target...**，选择 **Broadcast Upload Extension**。
2. 在弹出的对话框中填写相关信息，不用勾选 **Include UI Extension**，单击 **Finish** 完成创建。
3. 将下载到的 SDK 压缩包中的 **TXLiteAVSDK\_ReplayKitExt.framework** 拖动到工程中，勾选刚创建的 Target。
4. 选中新增加的 Target，依次单击 **+ Capability**，双击 **App Groups**，如下图：



操作完成后，会在文件列表中生成一个名为 Target.entitlements 的文件，如下图所示，选中该文件并单击 + 号填写上述步骤中的 App Group 即可。



- 选中主 App 的 Target，并按照上述步骤对主 App 的 Target 做同样的处理。
- 在新创建的 Target 中，Xcode 会自动创建一个名为 "SampleHandler.swift" 的文件，用如下代码进行替换。需将代码中的 APPGROUP 改为上文中的创建的 App Group Identifier。

```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt

let APPGROUP = "group.com.tencent.comm.trtc.demo"

class SampleHandler: RPBroadcastSampleHandler,
TXReplayKitExtDelegate {

    let recordScreenKey =
Notification.Name.init("TRTCRecordScreenKey")

    override func broadcastStarted(withSetupInfo setupInfo:
[String : NSObject]?) {
        // User has requested to start the broadcast. Setup info
        from the UI extension can be supplied but optional.
```

```
        TXReplayKitExt.sharedInstance().setup(withAppGroup:
APPGROUP, delegate: self)
    }

    override func broadcastPaused() {
        // User has requested to pause the broadcast. Samples will
stop being delivered.
    }

    override func broadcastResumed() {
        // User has requested to resume the broadcast. Samples
delivery will resume.
    }

    override func broadcastFinished() {
        // User has requested to finish the broadcast.
        TXReplayKitExt.sharedInstance().finishBroadcast()
    }

    func broadcastFinished(_ broadcast: TXReplayKitExt, reason:
TXReplayKitExtReason) {
        var tip = ""
        switch reason {
        case TXReplayKitExtReason.requestedByMain:
            tip = "屏幕共享已结束"
            break
        case TXReplayKitExtReason.disconnected:
            tip = "应用断开"
            break
        case TXReplayKitExtReason.versionMismatch:
            tip = "集成错误 (SDK 版本号不符合)"
            break
        default:
            break
        }

        let error = NSError(domain:
NSStringFromClass(self.classForCoder), code: 0, userInfo:
[NSLocalizedFailureReasonErrorKey:tip])
        finishBroadcastWithError(error)
    }
}
```

```
override func processSampleBuffer(_ sampleBuffer:
CMSampleBuffer, with sampleBufferType: RPSampleBufferType) {
    switch sampleBufferType {
    case RPSampleBufferType.video:
        // Handle video sample buffer
        TXReplayKitExt.sharedInstance()
        .sendVideoSampleBuffer(sampleBuffer)
        break
    case RPSampleBufferType.audioApp:
        // Handle audio sample buffer for app audio
        break
    case RPSampleBufferType.audioMic:
        // Handle audio sample buffer for mic audio
        break
    @unknown default:
        // Handle other sample buffer types
        fatalError("Unknown type of sample buffer")
    }
}
}
```

**步骤3:** 在组件中使用您的 App Group 和 Broadcast Upload Extension name 替换我们应用的 App Group 和 Broadcast Upload Extension name，代码位置请点击 [文件链接](#)。



```
void _startScreenSharing() {
    if (_isOffSeatInSeatMode()) {
        return;
    }

    if (_store.isSharing.value &&
        _store.screenShareUser.userId.value !=
        _store.currentUser.userId.value) {
        makeToast(
            msg: 'otherUserScreenSharing'.roomTr,
        );
        return;
    }

    String appGroup = '';
    if (Platform.isIOS) {
        appGroup = 'com.tencent.TUIRoomTXReplayKit-Screen';
        ReplayKitLauncher.launchReplayKitBroadcast('TXReplayKit_Screen');
    }
    _engineManager.startScreenSharing(appGroup: appGroup);
}
```

**步骤4：** 点击屏幕分享按钮，开始分享您的屏幕。

## 关键代码

若您想自定义实现屏幕共享功能，请参见 TUIRoomEngine SDK：[Android](#)、[iOS&Mac](#)、[Flutter](#)。

### 📌 说明：

如果您在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。