

iOS SDK 集成指南

- [iOS SDK 集成指南](#)

- - [1、修订记录](#)
 - [2、概述](#)
 - [3、简介](#)
 - [4、面向的读者](#)
 - [5、获取SDK Demo](#)
 - [6、技术支持](#)
 - [7、编写说明](#)
 - [8、集成步骤](#)

- - [a. 点击在 Xcode 工程左侧资源管理器中的工程图标。在右侧菜单 TARGETS -> General -> Link Frameworks and Libraries 的路径里, 点击“+”号增加系统框架。如下图所示:](#)
 - [b、工程需要用到的具体框架如下图:](#)
 - [c、导入库文件](#)
 - [d、创建全局头文件](#)
 - [e、导入 SDK API 头文件 avd_sdk.framework 下面的 avd_sdk.h 已经包含了全部功能模块的头文件导入, 只需要在步骤d中创建的全局头文件中导入一次 #import <avd_sdk/avd_sdk.h> 即可](#)

- [9、SDK Demo工程导入以及运行](#)

- - [a、在步骤5中下载的SDK Demo, 通过如下截图导入:](#)
 - [b、SDK Demo工程运行](#)
 - [c、点击运行按钮\(快捷键: command+R\) 开始运行demo代码, 截图如下:](#)
 - [d、 AppleID 登录 \(Xcode -> Preferences -> Accounts -> + \), 截图及步骤如下:](#)
 - [e、如果运行报错请登录自己 AppleID 使用 个人开发者 选择 个人开发证书, 截图如下:](#)

- [10、音视频基本功能开发流程](#)

- - [a、获取授权信息](#)
 - [b、目前 SDK Demo 提供的测试的授权信息及服务器地址如下 \(ps: 如需修改请在demo中 N2Meeting.m 文件中修改\), 公司会不定期的更新, 有问题请联系文档中的技术支持工程师咨询。](#)
 - [c、引擎初始化并设置相关代理](#)
 - [d、初始化引擎回调并设置相关视频默认分辨率 视频编码格式 \(h264 vp8\) 网络连接方式 \(tcp udp\)](#)
 - [e、引擎初始化成功之后就可以执行创建房间或者加入房间](#)
 - [f、加入房间相关代码以及回调](#)

- [g、加入房间成功之后通过room对象获取每个对应模块并设置对应代理](#)
- [h、本地开启/关闭音视频](#)
- [i、其他参会者加入/离开相关回调如下](#)
- [j、其他参会者打开/关闭摄像头回调以及自己本地摄像头开关回调](#)
- [k、其他参会者打开/关闭麦克风回调以及自己本地麦克风开关回调](#)
- [m、其他参会者视频订阅/取消订阅](#)

1、修订记录

修订日期	描述	作者	版本号
2019.6.3	初始文档	张启金	3.0.0

2、概述

- Open-AVD SDK 提供人与人实时沟通协作过程中需要用到的所有基本能力，涵盖了网络会议系统、IM即时通讯系统及直播系统三大类终端产品音视频通讯的主要功能。
- Open-AVD SDK 由业界资深工程师精心打造，稳定可靠，第三方团队拿来就能用，不必自己去造“轮子”，从而降低了第三方团队的技术风险，减少了项目的开发投入，尤其是能大幅缩短第三方团队开发具有多方音视频+数据协作能力的App/Web应用的时间。
- Open-AVD SDK 可用于几乎所有行业，很多业务场景中需要用到人与人实时沟通与协作的能力，而类似QQ，微信或会议系统这种通用沟通工具又不能直接使用或不能满足功能，这种情况下，Open-AVD SDK 就是您最好的选择。市场调研表明，Open-AVD SDK 在医疗、教育、金融、能源、交通等各个领域，都有巨大的市场需求。

3、简介

Open-AVD SDK 为移动、桌面和互联网应用提供一个完善的音视频及数据互动开发框架，屏蔽掉互动系统的复杂细节，对外提供较为简洁的 API 接口，方便第三方应用快速集成互动功能。

Open-AVD SDK iOS版提供如下功能：

- 房间管理服务
- 实时高清语音通话
- 实时高清视频通话
- 手机桌面共享功能
- 白板功能

4、面向的读者

本指南是提供给具有一定的iOS编程经验和了解面向对象概念的产品经理及程序员使用，

Open-AVD SDK 已很好的封装了音视频相关的底层技术细节，因而读者不需要具备音视频开发方面的经

验。

5、获取SDK Demo

公司的[github网址](#)上会提供各类基于 `Open-AVD SDK` 的实例 Demo，包括基本音视频能力Demo，桌面共享Demo，白板Demo等。

6、技术支持

您在使用本SDK的过程中，遇到任何困难，请与我们联系，我们将热忱为您提供帮助。您可以通过如下方式与我们取得联系。

技术支持工程师：微信号：`shanksJob`（岳成）

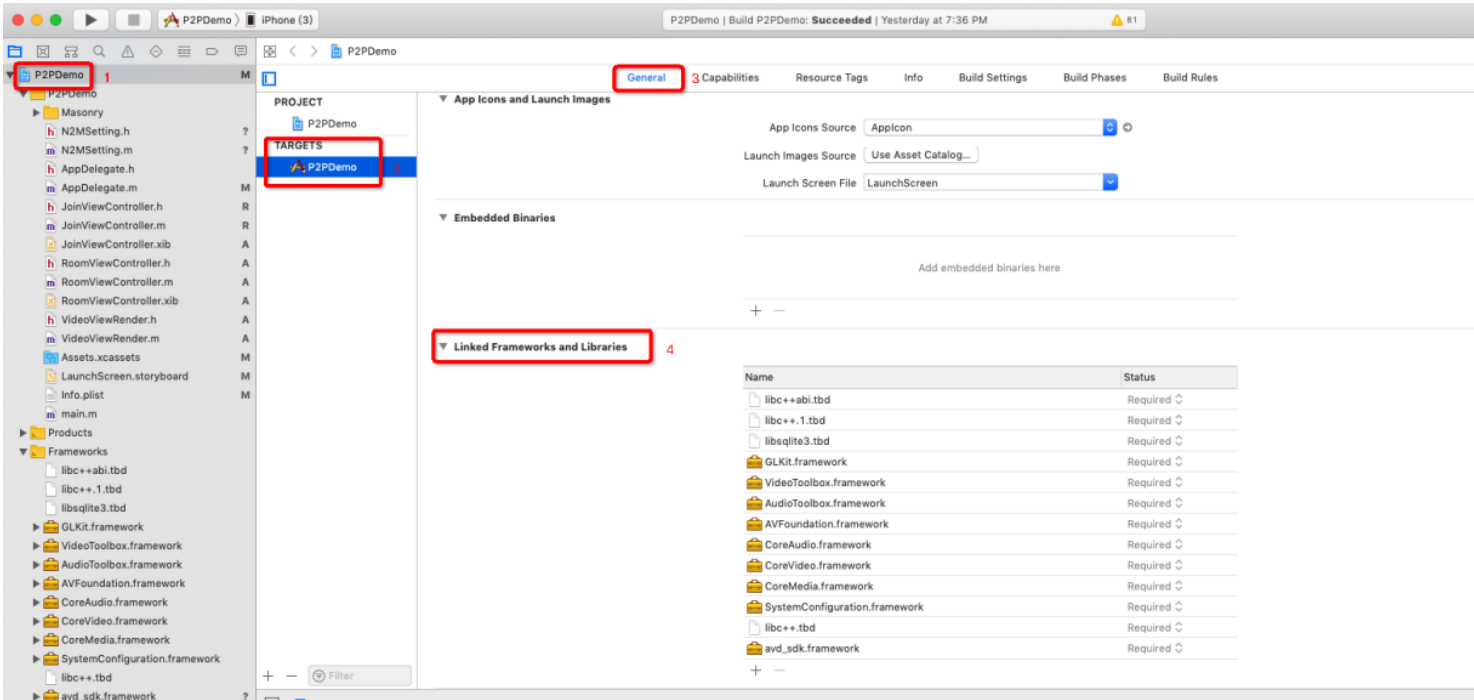
7、编写说明

本指南编写目的是为了帮助使用 `AVD SDK` 的用户快速搭建SDK的开发环境、熟悉开发流程、掌握SDK开发功能接口而编写的。

本指南基于iOS开发 `IDE Xcode9.0` 及以上，导入baseVideo Demo进行最简单的音视频能力进行编写，如果需要更多的功能，请参阅SDK API接口开发手册。

8、集成步骤

a. 点击在 `Xcode` 工程左侧资源管理器中的工程图标。在右侧菜单 `TARGETS` -> `General` -> `Link Frameworks and Libraries` 的路径里，点击“+”号增加系统框架。如下图所示：



b、工程需要用到的具体框架如下图：

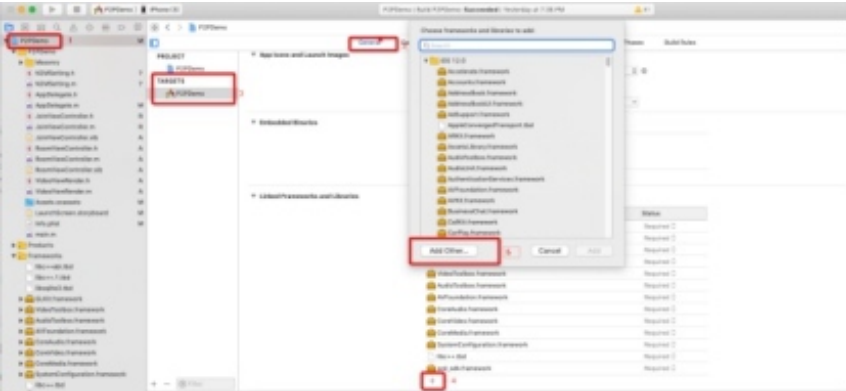
▼ Link Binary With Libraries (15 items)

Name	Status
ReplayKit.framework	Required ↕
Photos.framework	Required ↕
libc++.tbd	Required ↕
libc++abi.tbd	Required ↕
libsqlite3.tbd	Required ↕
GLKit.framework	Required ↕
VideoToolbox.framework	Required ↕
AudioToolbox.framework	Required ↕
avd_sdk.framework	Required ↕
AVFoundation.framework	Required ↕
CoreAudio.framework	Required ↕
CoreVideo.framework	Required ↕
CoreMedia.framework	Required ↕
SystemConfiguration.framework	Required ↕
libc++.tbd	Required ↕

+ — Drag to reorder frameworks

c、导入库文件

在添加开发框架步骤的同一处，点击 `Add other` 选项导入 `avd_sdk.framework` 包，该framework 开发包在 `sdk/` 目录里。如下图所示

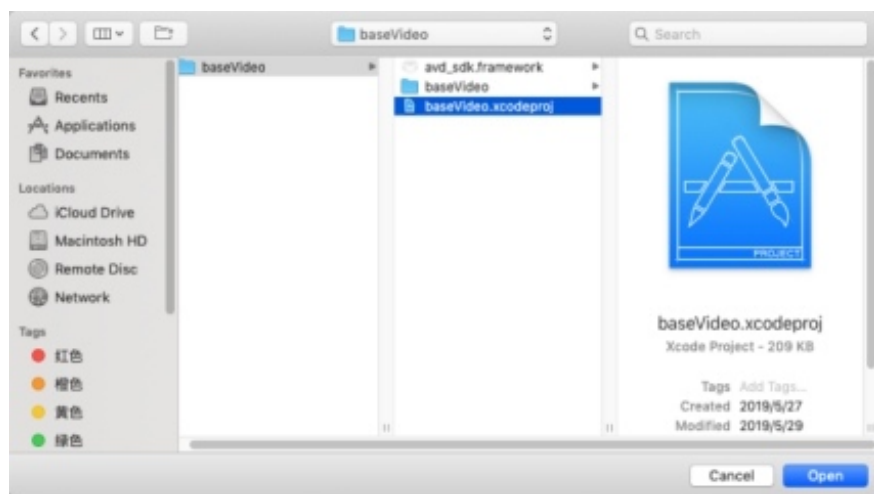
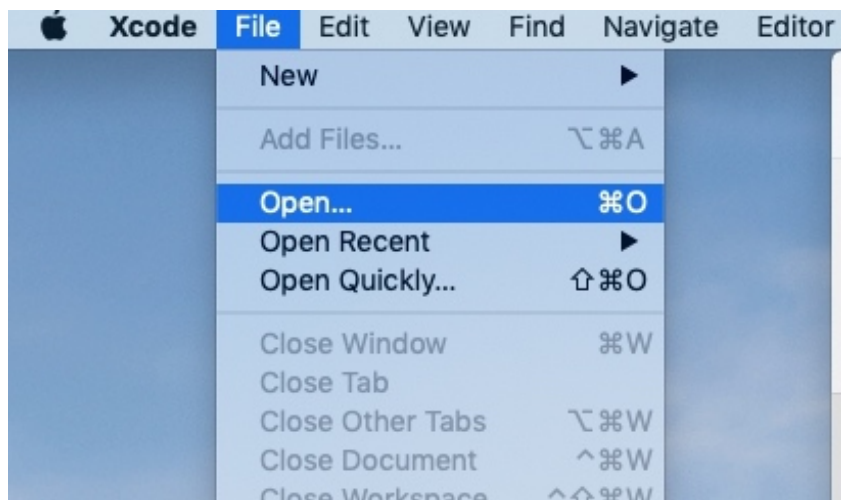


d、创建全局头文件

e、导入 `SDK API` 头文件 `avd_sdk.framework` 下面的 `avd_sdk.h` 已经包含了全部功能模块的头文件 导入，只需要在步骤d中创建的全局头文件中导入一次 `#import <avd_sdk/avd_sdk.h>` 即可

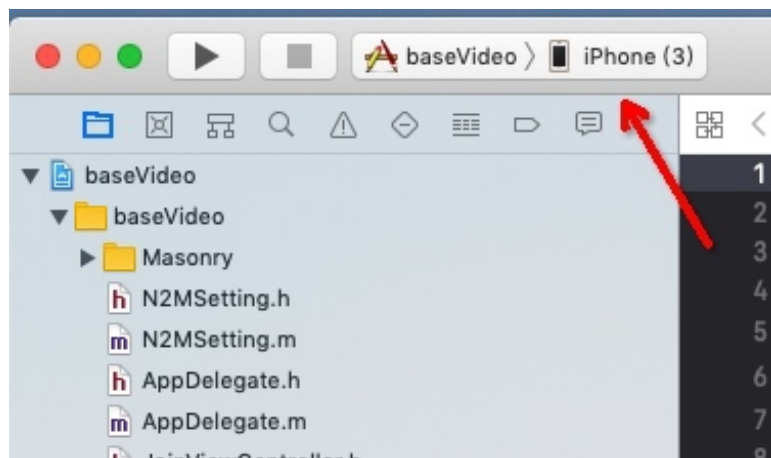
9、SDK Demo工程导入以及运行

a、在步骤5中下载的SDK Demo，通过如下截图导入：

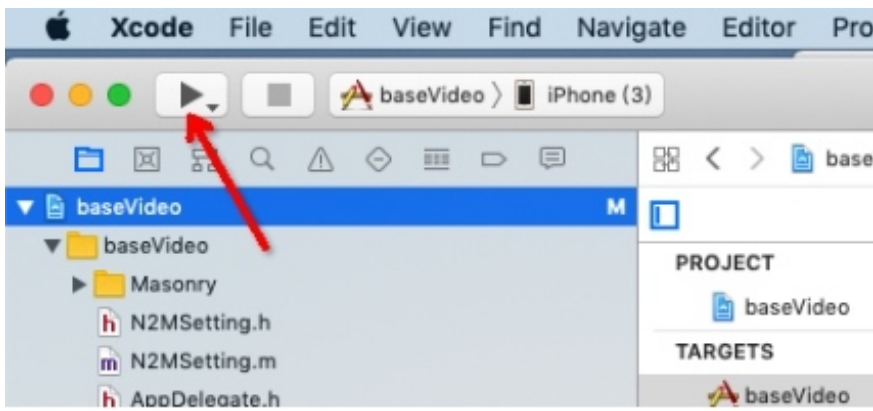


b、SDK Demo工程运行

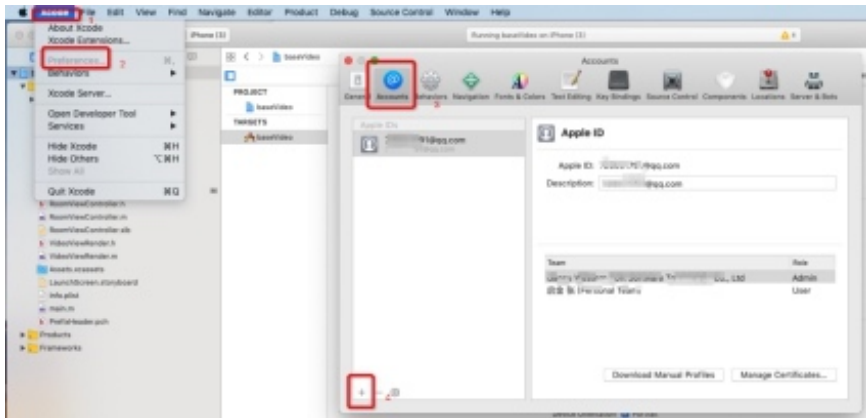
在SDK Demo成功导入工程后，选择真机调试，截图如下：



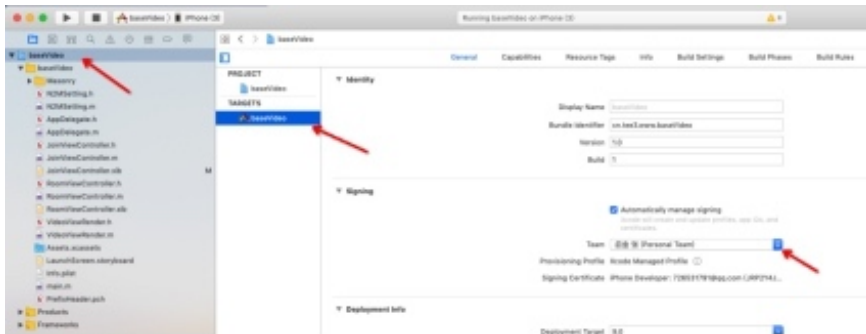
c、点击运行按钮（快捷键：`command+R`）开始运行demo代码，截图如下：



d、AppleID 登录（Xcode -> Preferences -> Accounts -> +），截图及步骤如下：



e、如果运行报错请登录自己 AppleID 使用 个人开发者 选择 个人开发证书，截图如下：



10、音视频基本功能开发流程

本开发流程只涉及到音视频的基本功能，在以baseVideo Demo的工程代码加以说明，关于 AVD SDK 其它的功能模块，如桌面共享、白板、用户管理等模块的集成开发，可以参阅 SDK API 接口开发手册或咨询我们的技术支持工程师。

a、获取授权信息

授权信息是针对 `AVD SDK` 的正式客户及测试客户，公司会提供一对有效的 `Access Key` 和 `Secret Key`，这对密钥可以产生访问令牌 `Token`（24小时有效），用于创建初始化SDK引擎成功之后生成会议房间号及加会房间等操作。

b、目前 `SDK Demo` 提供的测试的授权信息及服务器地址如下（ps：如需修改请在demo中 `N2Meeting.m` 文件中修改），公司会不定期的更新，有问题请联系文档中的技术支持工程师咨询。

```
serverUrl = "https://dev.3tee.cn:441";
accessKey = "SDKdemo_access";
secretKey = "SDKdemo_secret";
```

c、引擎初始化并设置相关代理

```
[[AVDEngine instance] initWithServerUrl:[N2MSetting getServerURL] accessKey:[N2MSetting get
AppKey] secretKey:[N2MSetting getSecretKey] delegate:self];
```

d、初始化引擎回调并设置相关视频默认分辨率 视频编码格式（h264 vp8） 网络连接方式（tcp udp）

```
- (void)onInitResult:(AVDResult)result {
if (AVD_Success == result) { //回调成功之后，需要默认设置如下几个参数,具体定义参考AVDEngine.h
    [[UIApplication sharedApplication] setIdleTimerDisabled:YES]; //设置屏幕不自动锁屏
    [[AVDEngine instance] setOption:video_codec_priority value:[N2MSetting getVideoCodecO
ption]]; //设置视频编码格式
    [[AVDEngine instance] setOption:camera_capability_default value:[N2MSetting getVideoR
esolutionOption]]; //设置本地视频发布分辨率
    [[AVDEngine instance] setOption:data_channel_tcp_priority value:[N2MSetting getDataCh
annelNetOption]]; //设置网络连接方式
    NSLog(@"AppDelegate, AVDEngine onInitResult init success");
} else {
    [N2MSetting showAlertWithTitle:@"提醒" message:@"引擎初始化失败，请检查网络或者key"
delegate:self cancelButtonTitle:@"知道了"];
    NSLog(@"AppDelegate, AVDEngine onInitResult init failed. result:%lu", (unsigned long)r
esult);
}
}
```

e、引擎初始化成功之后就可以执行创建房间或者加入房间

```
//创建房间按钮
- (IBAction)creatRoom:(UIButton *)sender {
    if (!self.roomId.text.length) {
        [N2MSetting showAlertWithTitle:@"警告" message:@"创建房间时房间号不能为空" delegate:
self cancelButtonTitle:@"知道了"];
    }
    AVDRoomInfo *roomInfo = [[AVDRoomInfo alloc] initWithRoomId:self.roomId.text roomName:self
.roomId.text appRoomId:self.roomId.text ownerId:self.roomId.text maxAttendee:5 maxAudio:5 max
Video:5 roomMode:1];
    [[AVDEngine instance] scheduleRoom:roomInfo];
}
```

创建房间回调

```
- (void)onScheduleRoomResult:(AVDResult)result roomId:(AVDRoomId)roomId{
    if (AVD_Success == result) {
        [N2MSetting showAlertWithTitle:@"创建成功" message:[NSString stringWithFormat:@"您
创建的房间号为: %@",roomId] delegate:self cancelButtonTitle:@"知道了"];
        self.roomId.text = roomId;
    }else {
        [N2MSetting showAlertWithTitle:@"失败" message:@"创建房间失败" delegate:self cancel
ButtonTitle:@"知道了"];
    }
}
```

f、加入房间相关代码以及回调

加入房间并设置相关代理（回调）参考代码：

```
- (void)joinRoom{
    NSString* roomId = self.roomId.text;
    if ([roomId length]>0&& [self.userName.text length] >0) {
        AVDRoom* room = [AVDRoom obtain:roomId];
        AVDUser* user = [[AVDUser alloc] initWithUserId:[N2MSetting getUserId] userName:self.
userName.text userData:nil];
        [room joinWithUser:user delegate:self];
    }else if ([self.roomId.text length]<=0){
        [N2MSetting showAlertWithTitle:@"提醒" message:@"房间号不能为空" delegate:self canc
elButtonTitle:@"知道了"];
    }else{
        [N2MSetting showAlertWithTitle:@"提醒" message:@"用户名不能为空" delegate:self canc
elButtonTitle:@"知道了"];
    }
}
```



```
}  
}
```

加入房间回调，参考代码如下

```
- (void)onJoinResult:(AVDResult)result{  
    NSLog(@"ViewController onJoinResult, result:%lu", (unsigned long)result);  
    if (AVD_Success == result) {  
        [N2MSetting setCurrentRoomId:self.roomId.text];  
        [N2MSetting setUsername:self.userName.text];  
        RoomViewController * roomVC = [[RoomViewController alloc] init];  
        roomVC.room = [AVDRoom obtain:self.roomId.text];  
        [self presentViewController:roomVC animated:YES completion:nil];  
    }else{  
        [N2MSetting showAlertWithTitle:@"警告" message:@"进入房间失败, 请检查账号是否正确" de  
legate:self cancelButtonTitle:@"知道了"];  
    }  
}
```

g、加入房间成功之后通过room对象获取每个对应模块并设置对应代理

```
self.mvideo = [AVDVideo getVideo:self.room]; //获取视频模块  
self.mvideo.delegate = self; //设置视频模块代理  
self.maudio = [AVDAudio getAudio:self.room]; //获取音频模块  
self.maudio.delegate = self; //设置音频模块代理  
self.mmanager = [AVDUserManager getUserManager:self.room]; //s获取用户管理模块  
self.mmanager.delegate = self; //设置用户管理回调
```

h、本地开启/关闭音视频

打开视频：

```
[self.mvideo publishLocalCamera];
```

关闭视频：

```
[self.mvideo unpublishLocalCamera];
```

打开音频：

```
[self.maudio openMicrophone];
```

关闭音频：

```
[self.maudio closeMicrophone];
```

i、其他参会者加入/离开相关回调如下

```

/** 用户加入房间通知
 *
 * @param[in] *user 用户信息数据。
 *
 * @note 某用户调用房间中的join加入房间后，房间内所有用户会接收到此通知
 * @sa join
 */
- (void)onUserJoinNotify:(AVDUser *)user;
/** 用户离开房间通知
 *
 * @param[in] *user 用户信息数据。
 *
 * @note 某用户调用房间中的leave离开房间后，房间内所有用户会接收到此通知
 * @sa leave
 */
- (void)onUserLeaveNotify:(AVDUser *)user;
//指示用户离开房间 reason == 807: 服务器端报807错误, 说明UDP不通或UDP连接超时 reason == 804: 同一个userid加会, 把前一个人踢下线 reason == 808: 调用rest api接口把人踢下线reason == 809: 无授权 15分钟以后强制踢人 其它: 你被踢出会议室
- (void)onLeaveIndication:(AVDResult)reason fromUser:(AVDUserId)fromId;

```

j、其他参会者打开/关闭摄像头回调以及自己本地摄像头开关回调

```

/** 获取已经打开的摄像头信息列表
 * @note 摄像头列表包含本地发布的摄像头和房间中其他用户的所有正在发布的摄像头，摄像头视频要在界面上显示，远端摄像头需要首先做subscribe；
 * 而本地摄像头显示时不需要subscribe，应用层开发时需要注意。publishedCameras中保存的是AVDCamera对象。
 */
@property (nonatomic,retain,readonly) NSMutableArray* publishedCameras;
/** 已订阅摄像头信息列表
 * @note subscribedCameras中保存的是AVDCamera对象。
 */
@property (nonatomic,retain,readonly) NSMutableArray* subscribedCameras;

/** 摄像头状态更改通知
 *
 * @param[in] status 摄像头状态。
 * @param[in] fromId 摄像头Id，唯一标示一个摄像头。
 * @note 当摄像头状态更改后，房间内所有用户接收到此通知。
 */
- (void) onCameraStatusNotify:(enum AVDDeviceStatus)status deviceId:(AVDDeviceId)fromId;
/** 摄像头视频发布通知

```

```

*
* @param[in] *camera 摄像头信息，摄像头信息中level和description为应用层数据，应用层可以使用这些字
段保存次摄像头视频相关的应用逻辑数据，方便标示、订阅视频。
*
* @note 当摄像头视频发布时，房间内所有用户接收到此通知；因通知的重要程度，将从onCameraStatusNotify
中分离出此状态。
* @sa publishLocalCamera
* @sa publishLocalAssist
*/
- (void) onPublishCameraNotify:(AVDCamera*)camera;
/** 摄像头视频取消发布通知
*
* @param[in] *camera 摄像头信息。
*
* @note 当摄像头视频取消发布时，房间内所有用户接收到此通知；因通知的重要程度，将从onCameraStatusNot
ify中分离出此状态。
* @sa unpublishLocalCamera
* @sa unpublishLocalAssist
*/
- (void) onUnpublishCameraNotify:(AVDCamera*)camera;
/// 异步返回
/** 本用户订阅视频异步返回
*
* @param[in] result 错误代码。
* @param[in] fromId 摄像头Id，唯一标示一路视频。
*
* @sa subscribe
*/
- (void) onSubscribeResult:(AVDResult)result deviceId:(AVDDeviceId)fromId;
/** 本用户取消订阅视频异步返回
*
* @param[in] result 错误代码。
* @param[in] fromId 摄像头Id，唯一标示一路视频。
*
* @sa unsubscribe
*/
- (void) onUnsubscribeResult:(AVDResult)result deviceId:(AVDDeviceId)fromId;
/** 本用户发布摄像头视频异步返回
*
* @param[in] result 错误代码。
* @param[in] fromId 摄像头Id，唯一标示一路视频。
*
* @sa publishLocalCamera
* @sa publishLocalAssist
*/

```

```

- (void) onPublishLocalResult:(AVDResult)result deviceId:(AVDDeviceId)fromId;
/** 本用户取消发布摄像头视频异步返回
 *
 * @param[in] result 错误代码。
 * @param[in] fromId 摄像头Id, 唯一标示一路视频。
 *
 * @sa unpublishLocalCamera
 * @sa unpublishLocalAssist
 */
- (void) onUnpublishLocalResult:(AVDResult)result deviceId:(AVDDeviceId)fromId;

```

k、其他参会者打开/关闭麦克风回调以及自己本地麦克风开关回调

```

/** 麦克风状态更改通知
 *
 * @param[in] status 麦克风状态。
 * @param[in] fromId 设备关联用户Id。
 * @note 当麦克风状态更改后, 房间内所有用户接收到此通知。
 * @sa openMicrophone
 * @sa closeMicrophone
 */
- (void) onMicrophoneStatusNotify:(enum AVDDeviceStatus)status fromUser:(AVDUserId) fromId;

@optional
/** 语音激励通知
 *
 * @param[in] info 语音激励信息。
 * @note 语音激励通知, 只有启用语音激励后才会有语音激励通知。启用语音激励接口为: monitorAudioLevel。
 * @sa monitorAudioLevel
 */
- (void) onAudioLevelMonitorNotify:(AVDAudioInfo*)info;

/// 异步返回
/** 本用户打开麦克风异步返回
 *
 * @param[in] result 错误代码。
 * @sa openMicrophone
 */
- (void) onOpenMicrophoneResult:(AVDResult)result;
/** 本用户关闭麦克风异步返回
 *
 * @param[in] result 错误代码。
 * @sa closeMicrophone

```

```
*/  
- (void) onCloseMicrophoneResult:(AVDResult)result;
```

m、其他参会者视频订阅/取消订阅

- 如果在自己加入房间之前已经有其他参会者在房间里面并且发布了视频则参考如下代码进行视频订阅操作

```
if (self.mvideo.publishedCameras.count>0) { //判断当前房间是否有人已经发布视频  
    __weak typeof(self) weakSelf = self;  
    //获取已经发布的视频流列表  
    [self.mvideo.publishedCameras enumerateObjectsUsingBlock:^(AVDCamera *camer, NSU  
Integer idx, BOOL * _Nonnull stop) {  
        __strong typeof(weakSelf) strongSelf = weakSelf;  
        if (![strongSelf.mvideo isSelfDevice:camer.id]) { //不是自己的视频才去订阅, ps  
: 自己视频发布之后默认订阅  
            [strongSelf.mvideo subscribe:camer.id]; //订阅  
        }  
    }  
}];  
}
```

- 如果在自己加入到房间之后在有其他参会者后加入房间里面并且发布了视频则参考如下代码进行视频订阅操作.

```
//房间中所有参会者发布视频都会走这个回调  
- (void) onPublishCameraNotify:(AVDCamera*)item {  
    if (![self.mvideo isSelfDevice:item.id]) {  
    }else {  
        AVDResult ret = [self.mvideo subscribe:item.id];  
    }  
}
```

- 订阅成功之后在对应回调中进行视频渲染操作

```
//视频流订阅回调  
- (void) onSubscribeResult:(AVDResult)result deviceId:(AVDDeviceId)fromId {  
    if (AVD_Success != result) {  
        return ;  
    }  
    [self.mvideo attachRenderWithDeviceId:fromId render:self.bigRender];  
}
```

- 在收到他人关闭摄像头通知之后进行取消他人视频订阅操作：

```
//房间中所有参会者取消发布视频都会走这个回调
- (void) onUnpublishCameraNotify:(AVDCamera*)item {
    if ([self.mvideo isSelfDevice:item.id]) {
        return ;
    }
    [self.mvideo unsubscribe:item.id];
}
```

- 在取消订阅回调中进行取消渲染操作

```
//取消订阅回调
- (void) onUnsubscribeResult:(AVDResult)result deviceId:(AVDDeviceId)fromId {
    [self.mvideo detachRenderWithRender:self.bigRender];
    [self.mvideo detachRenderWithDeviceId:item.id];
}
```

- 自己本地视频流渲染操作

```
//自己发布本地视频回调
- (void) onPublishLocalResult:(AVDResult)result deviceId:(AVDDeviceId)fromId {
    [self.mvideo attachRenderWithDeviceId:fromId render:self.localRender];
}
```

- 自己本地视频流取消渲染操作

```
//自己发布本地视频回调
- (void) onUnpublishLocalResult:(AVDResult)result deviceId:(AVDDeviceId)fromId {
    [self.mvideo detachRenderWithRender:fromId render:self.localRender];
    [self.mvideo detachRenderWithDeviceId:fromId];
}
```

- 离开房间并清理相关对象

```
- (void)leaveRoom{
    [self.room leave:0];
    [self.room destory];
}
```

```
self.room = nil;
self.mUserManager = nil;
self.mAudio = nil;
self.mVideo = nil;
}
```

- 关闭房间并清理相关对象

```
- (void)closeRoom{
    [self.room close];
    [self.room destory];
    self.room = nil;
    self.mUserManager = nil;
    self.mAudio = nil;
    self.mVideo = nil;
}
```

名词	解释	备注
AVDRoom	房间对象	是实时沟通功能的一个管理单元，房间中会有多个沟通参与者即用户，房间有各种沟通功能，如文字聊天、语音视频等，沟通是基于房间的。不同房间沟通是隔离的
AVDUser	用户对象	每个加入到房间的客户端作为一个房间用户，用户将会根据权限和设备情况执行房间中各种沟通功能
userId	用户id	唯一标示一个房间用户的Id，由应用层来设置
AVDVideo	视频模块	定义了视频的发布、订阅功能接口，摄像头打开、关闭回调、视频预览功能、获取已发布的视频链表、已订阅的视频链表等
AVDAudio	音频模块	定义了房间中音频相关如麦克风、扬声器的通知和异步操作回调

AVDUserManager	用户管理模块	对同一个房间的参会者的离开和加入回调通知