Agora Native SDK for iOS

参考手册

1.5 版

support@agora.io

目录

简介

Agora 实时云

Agora 通信即服务（Agora Communications as a Service, CaaS）运用 Agora 全球网络（Agora Global Network）为客户提供基于互联网的音视频通信，在实时通信和移动对移动通信方面进行特别优化，确保满意的用户体验质量。Agora 实时云致力于解决移动设备的用户体验问题、3G/4G/Wi-Fi 网络性能各异、全球网络瓶颈等一系列问题，为用户带来优质的通信体验。

Agora 实时云包含以下 SDK，且 Native SDK 在程序生成时直接与应用程序建立连接:

- **Agora Native SDK for iOS** 和 **Agora Native SDK for Android**

  用于移动设备的 Agora Native SDK，专为智能手机研发，其通信基于 Agora Global Network，并针对不同移动设备平台进行优化。

  在下文的入门指南一节中将为您阐述如何使用 Agora Native SDK 组件为 iOS 平台搭建项目环境。

- **Agora Native SDK for Windows** 和 **Agora Native SDK for Mac**

- **Agora Native SDK for Web**

  针对网页通信开发，为网页浏览器提供 Agora Global Network 通信支持。

- **Agora Whiteboard SDK**

  为全平台提供白板功能，供用户画图、注释、上传文件，方便沟通。

Agora Native SDK for iOS

Agora Native SDK for iOS 可编程实现以下功能：

- **建立通话**：加入或离开会话（会话以频道名称为标识），既可实现全球用户多方会话，也可是一对一的聊天对话。频道名称应为唯一的，由应用程序代码创建及维护，通常由用户、会议和时间信息组成。
- **音视频控制**：启用和禁用音视频、静音设置、并可设置多种音视频参数以达到通话最佳效果。Agora SDK 的大部分操作均自动设置为默认值，因此即使未给参数赋值也可运行。
- **设备控制**：访问麦克风或话筒、音量设置、选择摄像头、设置视频显示窗口等。
- **通话管理**：当其他用户加入或离开通话（频道）时可接收事件通知，可获知通话对方是谁，静音的用户是谁以及哪个用户在观看视频通话等等。这些事件通知有助于应用程序决定是否对音视频流做调整，或修改用户及状态信息等。

- **质量管理**：获取网络质量统计数据、运行内嵌式测试、提交通话质量评分和投诉，以及启用各级错误记录等。
- **音视频录制**：同时录制同一频道内或多个频道内的语音视频会话。该功能仅适用于使用了动态密钥的用户。
- **数据加密**：对语音和视频数据进行加密。加密频道无法使用录制功能。

Agora Native SDK for iOS 提供多个 Objective-C 类库实现以下功能：

- `AgoraRtcEngineKit` 接口类包含所有由应用程序调用的主要方法。
- `AgoraRtcEngineDelegate` 接口类采用 delegate 方法用于向应用程序发送回调通知。

关于以上所有类及其方法的详细说明，见章节 Agora Native SDK for iOS - API 参考。

在 Agora SDK 调用 API 或运行时，可能会返回错误或警告代码，详见附录。

用户须知

搭建项目环境要求

- Apple XCode 6.0 或以上版本
- 支持语音功能的 iOS 模拟器或真机设备

注：只有真机设备支持视频功能

所需库

Agora Native SDK 需要 iOS 6.0 SDK 或更新版本的 iOS SDK。主项目应当与 SDK 中的下列库有连接：

- AgoraRtcEngineKit.framework
- AudioToolbox.framework
- VideoToolbox framework
- AVFoundation.framework
- CoreMedia.framework
- CoreTelephony.framework
- CoreMotion.framework
- SystemConfiguration.framework
- libc++.dylib

注：Agora Native SDK 默认使用 libc++ (LLVM)，如需使用 libstdc++ (GNU)，请联系 support@agora.io。

在源文件中，使用下列命令导入 `AgoraRtcEngineKit`：

- `#import <AgoraRtcEngineKit/AgoraRtcEngineKit.h>`

SDK 提供的库是 Fat Image，包含 32/64 位模拟器、32/64 位真机版本。

所需文档

通过获取 SDK 下载 SDK 后，请阅读 SDK 内以下文档：

- 本参考手册：必读
- Agora Recording Server 用户指南：如需使用录制功能，请结合本参考手册一起使用。
- 数据加密用户指南：如需使用数据加密功能，请结合本参考手册一起使用。

入门指南

获取 SDK

请登录 http://cn.agora.io/download/ 网站下载 **Agora Native SDK for iOS**，或联系 sales@agora.io 获取最新版本。

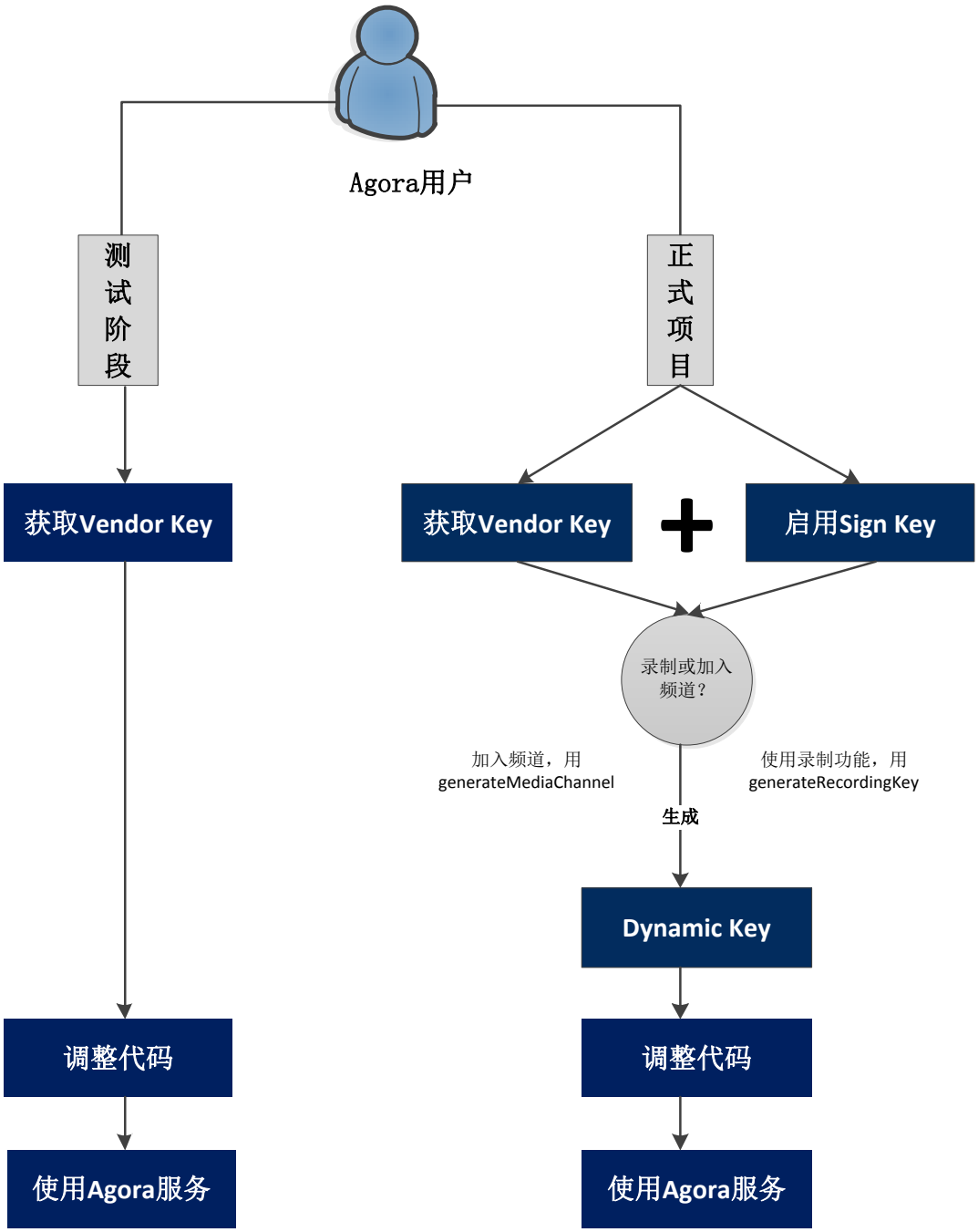
设置密钥方案

您在使用 Agora SDK 时可能需要用到多种密钥，本章主要介绍 Vendor Key, Sign Key 和 Dynamic Key 的概念和用法。

密钥介绍

每个在 Agora 官网注册的账户均能创建多个项目，且每个项目均有唯一的 Vendor Key。在 Agora Global Network 上的通信按照 Vendor Key 隔离，持有不同 Vendor Key 的用户即使加入的频道名称相同也会隔离，不会互相干扰。但 Vendor Key 是一种静态密钥，如果有人非法获取了您的 Vendor Key，他将可以在 Agora 提供的 SDK 中使用您的 Vendor Key，如果他知道您的频道名字，甚至有可能干扰您正常的通话。

Dynamic Key 提供更为安全的用户身份验证方案，用户每次访问 Agora 的服务时(例如加入频道或使用录制功能时)，后台服务通过 Agora 提供的 Vendor Key 和 Sign Key,基于 HMAC 的安全算法生成一个新的动态密钥发送给客户端，客户端调用加入频道或使用录制功能的 API 接口函数使用此动态密钥。Agora 的服务器通过动态密钥验证用户是否合法。



注：

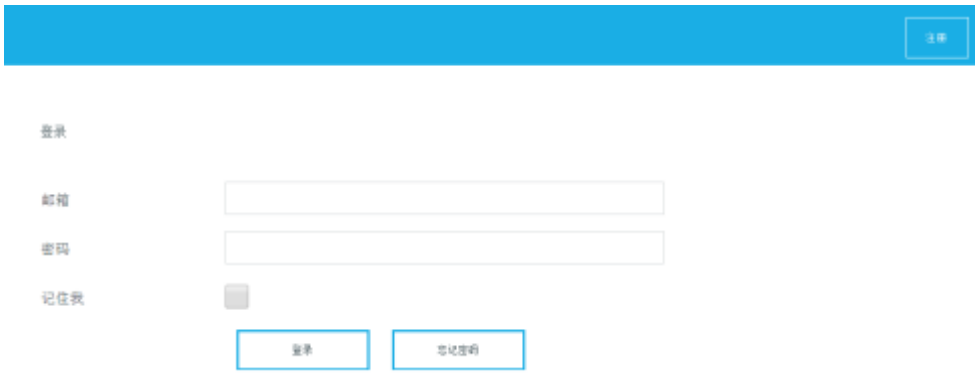
- 图为Agora建议的用法，测试阶段仅使用Vendor Key可以方便您快速搭建项目，正式项目建议使用Dynamic Key提高安全性。
- 如需使用录制功能，必须使用Dynamic Key。
- Sign Key仅用来生成Dynamic Key，不可单独使用。

获取和使用 Vendor Key

获取 Vendor Key

每个 Agora 账户下可创建多个项目，且每个项目有独立的 Vendor Key。

1. 访问 https://dashboard.agora.io/ ，点击右上角的**注册**，如下图所示：



2. 在**项目列表**页面点击**新增**。
3. 填写**项目名**，如不需使用动态密钥，则不需选择**启用 Sign Key**。点击**保存**。
4. 在所创建的项目下查看 Vendor Key。获取 Vendor Key 后即可使用对应的 Agora.io 服务功能。

使用 Vendor Key

使用每个项目下唯一的 Vendor Key 访问对应的 Agora.io 服务功能：

1. 在示例程序中启用音频或视频通话时，在启动窗口中输入您的 vendor key。
2. 在开发程序时，将 Vendor Key 嵌入您的代码中以便调用 SDK。
3. 当调用 sharedEngineWithVendorKey()进行初始化时,将参数 *vendorKey* 设为您的 Vendor Key。
4. 当调用 joinChannelByKey()加入频道时，将参数 *key* 设为 NULL。

获取和使用 Dynamic Key

获取 Vendor Key 和 Sign Key

每个 Agora 账户下可创建多个项目，且每个项目有独立的 Vendor Key 和 Sign Key。

根据以下步骤创建一个项目，同时获取 Vendor Key 和 Sign Key。

1. 登录 https://dashboard.agora.io。
2. 在**项目列表**页面点击**新增**。
3. 填写**项目名**，选择**启用 Sign Key**。点击**保存**。

   项目添加成功。

4. 查看已添加项目的 Vendor Key 和 Sign Key。

   点击锁标志显示 Sign Key，再次点击该标志隐藏 Sign Key。

如果您在创建项目时没有选择启用 Sign Key，在项目添加之后，可以点击图中所示的**启用 Sign Key** 获取 Sign Key。

注:

- 如果出于某种原因您需要更新 Sign Key，请联系 support@agora.io 。
- 将您的 Sign Key 保存在服务器端，且对任何客户端均不可见。
- Sign Key 的用法，详见表 1。

集成动态密钥算法

开发者将生成动态密钥的算法集成到自己企业的信令服务中。Agora 提供了示例代码，涵盖 C++, Java, Python, node.js 等语言，可以直接将代码应用在您的程序中。加入频道和使用录制功能的 Vendor Key 和 Sign Key 相同，但是生成 Dynamic Key 的方法不同，前者使用 generateMediaChannelKey，而后者使用 generateRecordingKey。

请登录以下任意网址获取示例代码：

- Dashboard 的 Sign Key 页面：https://dashboard.agora.io/signkey
- Github：https://github.com/AgoraLab/AgoraDynamicKey

如需要验证用户 ID(User ID), 详见下表所列的动态密钥与 SDK 的版本兼容：

动态密钥版本	User ID(UID)	SDK 版本
DynamicKey4	指定用户的 UID	1.3 或更高版本
DynamicKey3	指定用户的 UID	1.2.3 或更高版本
DynamicKey	N/A	N/A

如不需验证 UID，则各版本 SDK 可使用任意 DynamicKey 版本，但建议升级至 DynamicKey4。

使用动态密钥

每一次客户端加入频道或开始录制前（如发起会话、接收到会话邀请）：

1. 客户端请求企业自己的服务器的信令服务授权。
2. 服务器端基于 Sign Key、Vendor Key、频道名称、当前时间戳，客户端用户 id，有效期时间戳等信息通过 Agora 提供的算法生成动态密钥，返回给授权的客户端应用程序。
3. 客户端应用程序调用 `joinChannelByKey()` 或 `startRecordingService()`，第一个参数要求为动态密钥。
4. Agora 的服务器接收到动态密钥信息，验证该通话是来自于合法用户，并允许访问 Agora Global Network。

   注：采用了动态密钥方案后，在用户进入频道时，Dynamic Key 将替换原来的 Vendor Key。

动态密钥在一段时间后会失效，应用程序获知密钥失效（即当 didOccurError 回调方法报告 ERR_DYNAMIC_KEY_TIMEOUT（109））时，需调用 renewChannelDynamicKey()方法进行更新。

动态密钥的安全性

采用动态密钥方案的企业，在用户加入频道或使用录制功能时都需要提供动态密钥，确保用户每次通话都经过信令服务器的验证。动态密钥采用 HMAC/SHA1 签名方案，提高了系统及通话的安全性。

动态密钥结构

下表中所有字段从前往后拼接，共 103 字节。

字段	类型	长度	说明
版本号	字符串	3	动态密钥算法的版本信息
签名	字符串	40	签名的 hex 编码。将 Sign Key 以及下列字段作为输入，通过 HMAC 计算和 hex 编码而成的 40 字节字符串： ✓ 服务类型：ASCII 可见字符串，由 Agora 提供。详见章节*服务类型*。 ✓ Vendor key： 32 位 vendor key 字符串。 ✓ Timestamp：动态密钥生成时的时间戳。 ✓ 随机数：32 位整数 hex 编码。随每个请求重新生成。 ✓ Channel：频道名称，用户自定义，不超过 64 字节。 ✓ 用户 id：客户端自定义的用户 id ✓ 服务过期时间：用户在频道内可以通话截止的时间戳
Vendor Key	字符串	32	Vendor Key

授权时间戳	数字	10	动态密钥生成时的时间戳，自 1970.1.1 开始到当前时间的秒数。授权 5 分钟内可以访问 Agora 服务。
随机数	整数	8	32 位整数 hex 编码。随每个请求重新生成。
服务过期时间戳	数字	10	用户使用 Agora 服务终止的时间戳，在此时间之后，将不能继续使用 Agora 服务（比如进行的通话会被强制终止）；如果对终止时间没有限制，设置为 0.

表 1

服务类型

服务	值	说明
通话服务	ACS	Agora 提供的音视频通信服务，例如当使用 joinChannel API 时，如果需要传入动态密钥，需要使用这个类型来生成动态密钥。
录制服务	ARS	Agora 提供的音视频录音录像服务，例如当使用 startRecordingService API 时，需要使用这个类型来生成动态密钥。

签名算法：HMAC/SHA1

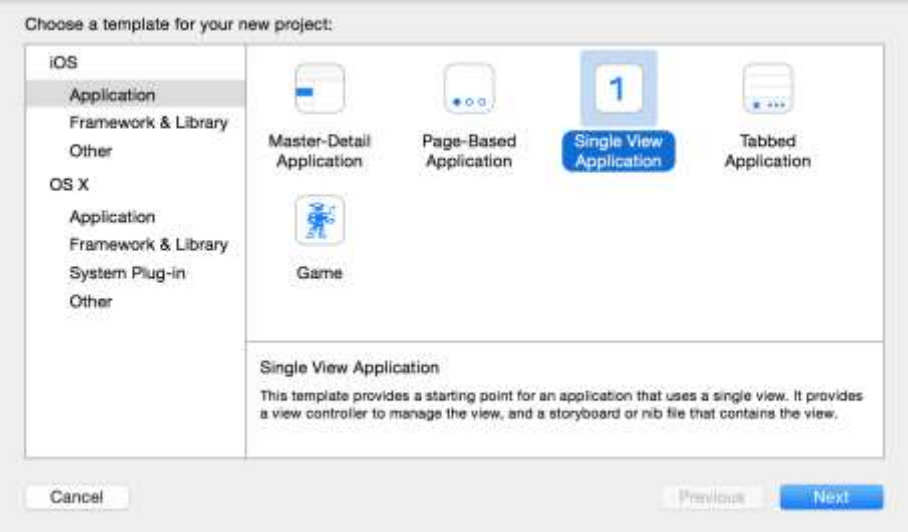
动态密钥采用业界标准化的 HMAC/SHA1 加密方案，在 Node.js, PHP, Python, Ruby 等绝大多数通用的服务器端开发平台上均可获得所需库。具体加密方案可参看以下网页：

http://en.wikipedia.org/wiki/Hash-based_message_authentication_code

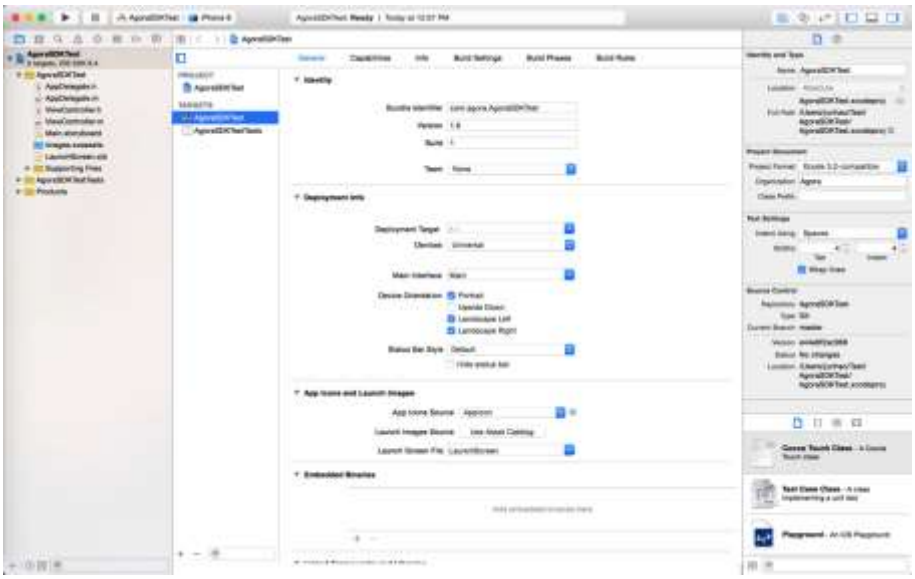
如需更多技术支持，请联系 Agora.io。

搭建项目

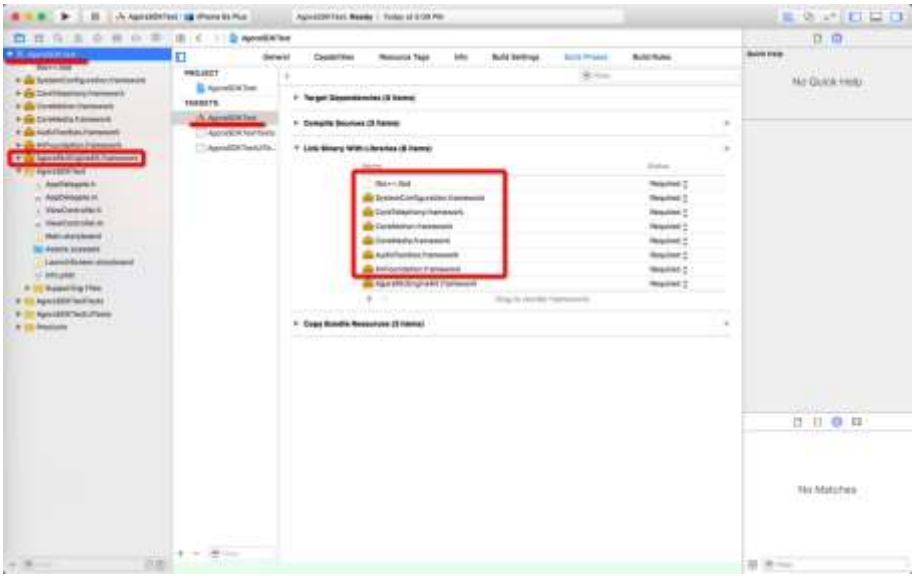
1. 创建项目，如下图所示。（注：下图仅以 Xcode 为示例）



项目创建完成后，如下图所示：



2. 在 **Agora Native SDK** 中找到 **libs\AgoraRtcEngineKit.framework**
3. 右键单击 **AgoraRtcEngineKit.framework**，从下拉菜单中选择 **Add Files to Project**，添加 **AgoraRtcEngineKit.framework** 到 project
4. **AgoraRtcEngineKit.framework** 需要一些系统 framework 的支持（相关 framework 见上述“所需库”一节），这些 framework 应在 Build Phase 中 **Link Binary With Libraries** 添加。如下图所示：



5. 使用`#import <AgoraRtcEngineKit/AgoraRtcEngineKit.h>`包含头文件，即可开始使用**Agora Native SDK for iOS**。

其余 **Agora Native SDK** 的方法，参见下文 *Agora Native SDK for iOS - API 参考*一节。

使用示例程序

在 Agora Native SDK for iOS 压缩包中可获得 AgoraDemo 示例程序:

AgoraDemo – 包含基本调用方法如加入和离开通话，演示 **Agora Native SDK for iOS** 的核心操作功能，采用简单方法调用即可启用音视频通话。

AgoraDemo/AgoraDemo：示例程序的源码文件夹

AgoraDemo/AgoraDemo.xcodeproj: 示例程序的项目文件

AgoraDemo/AgoraDemoTests: 示例程序的测试文件夹

libs/AgoraRtcEngineKit.framework: SDK 框架文件

注：如需更多示例程序的开源代码，请参阅网页：https://github.com/AgoraLab/。

编译示例程序

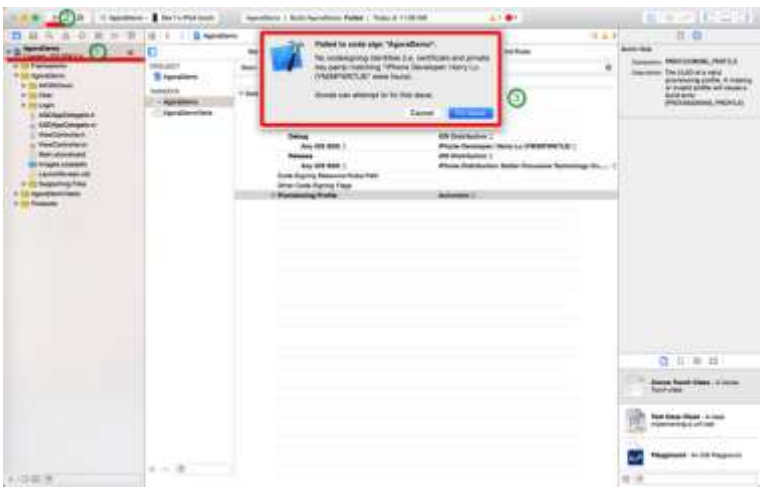
1. 运行 XCode
2. 打开 AgoraDemo.xcodeproj 文件

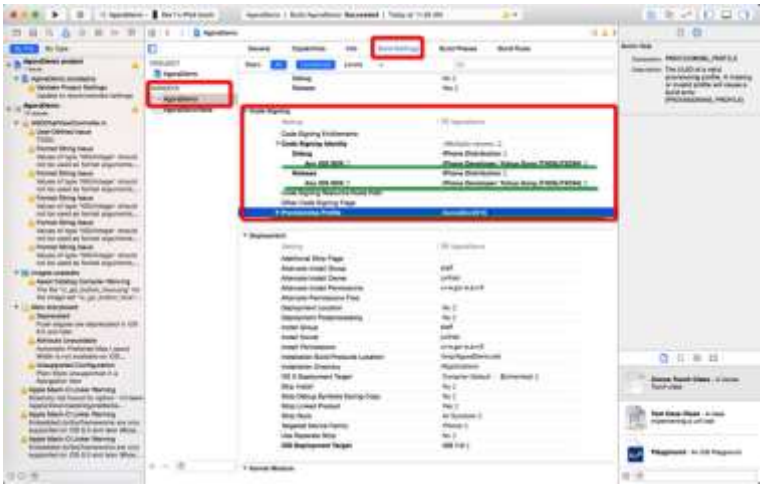
3. 打开项目文件后，显示下图对话框。左侧面板显示 AgoraDemo 的文件结构，如下图红框内所示：

4. 选中项目，如图中#1 所示，点击 build and run 按钮进行编译，如#2 所示。

注： a) AgoraDemo 只支持真机，不支持模拟器；

b) 编译中如果出现下图中#3 的错误提示，请将 code signing 修改为自己的激活设备的选项，或者应用自己的 apple 开发者账号。





运行示例程序

编译完成后，即可运行 AgoraDemo 示例程序，第一个页面如下图所示。演示 AgoraDemo 需要两部手机，其功能包括语音通话和视频通话

1. 演示语音通话功能：

    a) 在两部手机上输入相同的 vendor key 和自定义的会议名称（即频道名称），下图以会议室“222222”为例：

b) 点击“进入语音通话”按钮，进入语音通话页面

c) 通话页面包含以下几个功能：静音/关闭静音、打开/关闭扬声器、离开会议

2. 演示视频通话功能：点击或轻按“视频通话”按钮，进入视频通话主页面。

a) 在两部手机上输入相同的 vendor key 和自定义的会议名称（即频道名称）。

b) 点击或轻按“加入视频通话”按钮，进入通话页面。

c) 通话页面包含以下几个功能：静音/关闭静音、打开/关闭扬声器、打开/关闭摄像头、切换摄像头、离开会议。

3. Agora SDK 允许用户在语音通话和视频通话之间转换，轻按屏幕底部“语音通话”或“视频通话”按钮进行切换。

注：Agora SDK 和 AgoreDemo 程序支持多方会话，最多可达五方会话，如下图所示。不同用户均需输入相同的 vendor key 和会议名称（即频道名称）。


Room 2804
Video Call
Voice Call

Agora Native SDK for iOS - API 参考

AgoraRtcEngineKit 接口类

AgoraRtcEngineKit 是 **Agora Native SDK** 的基本接口，通过生成 AgoraRtcEngineKit 实例，调用该实例的方法使用 **Agora Native SDK** 的通话能力。在之前的版本中，该类名为 AgoraAudioKit，从 1.0 版本更改为 AgoraRtcEngineKit。

初始化 (sharedEngineWithVendorKey)

```
(id)
(instancetype)sharedEngineWithVendorKey:(NSString*)vendorKey
delegate:( id<AgoraRtcEngineDelegate>) delegate;
```

以上代码初始化 AgoraRtcEngineKit 为一个单例。使用 AgoraRtcEngineKit，必须先调用该接口进行初始化。

Agora Native SDK 通过指定的 delegate 通知应用程序引擎运行时的事件。Delegate 中定义的所有方法都是可选实现的。

名称	描述
vendorKey	Agora 为应用程序开发者签发的厂商秘钥。如果没有，请向 Agora 申请。
delegate	

开启视频模式 (enableVideo)

```
(int)enableVideo
```

该方法用于开启视频模式。可以在加入频道前或者通话中调用，在加入频道前调用，则自动开启视频模式，在通话中调用则由音频模式切换为视频模式。使用 disableVideo 方法可关闭视频模式。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

开启纯音频模式 (disableVideo)

```
(int)disableVideo
```

该方法用于关闭视频，开启纯音频模式。可以在加入频道前或者通话中调用，在加入频道前调用，则自动开启纯音频模式，在通话中调用则由视频模式切换为纯音频频模式。使用 enableVideo 方法可开启视频模式。

名称	描述
Return Value	0：方法调用成功

	<0：方法调用失败

开启视频预览(startPreview)

(int)startPreview

该方法用于启动本地视频预览。在开启预览前，必须先调用 setupLocalVideo 设置预览窗口及属性，且必须调用 enableVideo 开启视频功能。如果在调用 joinChannelByKey 进入频道之前调用了 startPreview 启动本地视频预览，在调用 leaveChannel 退出频道之后本地预览仍然处于启动状态，如需要关闭本地预览，需要额外调用 stopPreview 一次。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

停止视频预览(stopPreview)

(int)stopPreview

该方法用于停止本地视频预览。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

加入频道 (joinChannelByKey)

(int)joinChannelByKey:(NSString *)key channelName:(NSString *)channelName info:(NSString *)info uid:(NSUInteger)uid joinChannelSuccess:(void(^)(NSString* channel, NSUInteger uid, NSInteger elapsed))joinChannelSuccessBlock;

该方法让用户加入通话频道，在同一个频道内的用户可以互相通话，多个用户加入同一个频道，可以群聊。使用不同 vendor key 的应用程序是不能互通的。如果已在通话中，用户必须调用 leaveChannel 退出当前通话，才能进入下一个频道。该方法是异步的，所以可以在主用户界面线程被调用。

注：SDK 在通话中使用 iOS 系统的 AVAudioSession 共享对象进行录音和播放，应用程序对该对象的操作可能会影响 SDK 的音频相关功能。在 joinChannel 时，SDK 调用 setCategory AVAudioSessionCategoryPlayAndRecord 将 AVAudioSession 设置到 PlayAndRecord 模式，应用程序不应将其设置到其他模式。设置该模式时，正在播放的声音会被打断（比如正在播放的响铃声）。

名称	描述
key	此为程序生成的动态 Key；

	当用户使用静态 Key 也即只使用 vendor key 时，该参数是可选的，传 **NULL** 即可； 当用户使用动态 Key 时，Agora 为应用程序开发者额外签发一个签名秘钥 sign key，开发者通过 Agora 提供的算法和秘钥生成此用户 Key，用于服务器端用户验证。 一般来说使用 Vendor Key 即可，对于安全有极高要求的使用者可使用动态密钥。对于使用录制功能的用户，必须使用动态密钥。
channelName	标识通话的频道名称，长度在 64 字节以内的字符串。 以下为支持的字符集范围（共 89 个字符）： a-z A-Z 0-9 空格 ! #$%& ()+, - :;<=. >? @[] ^_` {|} ~
info	（非必选项）开发者需加入的任何附加信息。一般可设置为空字符串，或频道相关信息。该信息不会传递给频道内的其他用户。
uid	（非必选项）用户 ID，32 位无符号整数。建议设置范围：1 到 (2^32-1)，并保证唯一性。如果不指定（即设为 0），SDK 会自动分配一个，并在 userJoinedBlock 回调方法中返回，App 层必须记住该返回值并维护，SDK 不对该返回值进行维护。
Return Value	0：方法调用成功 <0：方法调用失败

离开频道 (leaveChannel)

```
(int)leaveChannel:(void(^)(AgoraRtcStats*
stats))leaveChannelBlock;
```

离开频道，即挂断或退出通话。joinChannel 后，必须调用 leaveChannel 以结束通话，否则不能进行下一次通话。不管当前是否在通话中，都可以调用 leaveChannel，没有副作用。leaveChannel 会把会话相关的所有资源释放掉。

leaveChannel 是异步操作，调用返回时并没有真正退出频道。在真正退出频道后，SDK 会触发 didLeaveChannelWithStats 回调。

名称	描述
leaveChannelBlock	成功离开频道的回调。
Return Value	0：方法调用成功 <0：方法调用失败

更新频道动态 key (renewChannelDynamicKey)

```
(int)renewChannelDynamicKey(NSString*) key;
```

该方法用于更新动态 Key。如果启用了动态 Key 机制，过一段时间后 Key 会失效。当 didOccurError 回调报告 AgoraRtc_Error_DynamicKeyTimeout(109)时，应用程序应重新获取 Key，然后调用该 API 更新 Key，否则 SDK 无法和服务器建立连接。

名称	描述
key	指定要更新的动态 Key。
Return Value	0：方法调用成功 <0：方法调用失败

获取通话 ID (getCallId)

```
(NSString*) getCallId;
```

获取当前的通话 ID，客户端在每次 joinChannelByKey 后会生成一个对应的 CallId，标识该客户端的此次通话。有些方法如 rate，complain 需要在通话结束后调用，向 SDK 提交反馈，这些方法必须指定 CallId 参数。使用这些反馈方法，需要在通话过程中调用 getCallId 方法获取 CallId，在通话结束后在反馈方法中作为参数传入。

名称	描述
Return Value	通话 ID

给通话评分 (rate)

```
(int) rate:(NSString*)callId rating:(NSInteger) rating
description(NSString*) description;
```

该方法能够让用户为通话评分，一般在通话结束后调用。

名称	描述
callId	通过 getCallId 方法获取的通话 ID
rating	给通话的评分，最低 1 分，最高 10 分。
description	给通话的描述，可选，长度应小于 800 字节。
Return Value	0：方法调用成功 <0：方法调用失败

	ERR_INVALID_ARGUMENT (-2)：传入的参数无效，比如 callId 无效。 ERR_NOT_READY (-3)：SDK 不在正确的状态，可能是因为没有成功初始化。

投诉通话质量 (complain)

(int) complain:(NSString*) callId description(NSString*) description;

该方法能够让用户就通话质量进行投诉，一般在通话结束后调用。

名称	描述
callId	通过 getCallId 方法获取的通话 ID
description	给通话的描述，可选，长度应小于 800 字节。
Return Value	0：方法调用成功 <0：方法调用失败 ERR_INVALID_ARGUMENT (-2)：传入的参数无效，比如 callId 无效。 ERR_NOT_READY (-3)：SDK 不在正确的状态，可能是因为没有成功初始化。

启动语音通话测试 (startEchoTest)

(int)startEchoTest :(void(^)(NSString* channel, NSUInteger uid, NSInteger elapsed))successBlock;

该方法启动语音通话测试，目的是测试系统的音频设备（耳麦、扬声器等）和网络连接是否正常。在测试过程中，用户先说一段话，在 10 秒后，声音会回放出来。如果 10 秒后用户能正常听到自己刚才说的话，就表示系统音频设备和网络连接都是正常的。

注：调用 startEchoTest 后必须调用 stopEchoTest 以结束测试，否则不能进行下一次回声测试，或者调用 joinChannel 进行通话。

名称	描述
successBlock	启动 echo test 成功回调，回调参数含义见 joinChannelByKey 的 joinSuccessBlock。
Return Value	0：方法调用成功 <0：方法调用失败 ERR_REFUSED (-5)：不能启动测试，可能没有成功初始化。

终止语音通话测试 (stopEchoTest)

(int)stopEchoTest;

该方法用于终止语音通话测试。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败 ERR_REFUSED(-5)：停止 EchoTest 失败，可能是因为不在进行 EchoTest。

启用网络测试 (enableNetworkTest)

```
(int)enableNetworkTest
```

该方法启用网络连接质量测试，用于检测用户网络接入质量。启用后，SDK 通过 networkQualityBlock 通知应用程序当前的网络质量。

注：启用后，在没有通话时也会消耗一些网络流量。建议的使用方法一般如下：

应用程序在前台时，调用 enableNetworkTest 启用网络接入检测；应用程序被切换到后台后，调用 disableNetworkTest 禁用检测以节省流量。网络测试默认为关闭状态。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

禁用网络测试 (disableNetworkTest)

```
(int)disableNetworkTest;
```

该方法禁用网络链接质量测试。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

静音 (muteLocalAudioStream)

```
(int)muteLocalAudioStream:(BOOL)muted;
```

开启/取消客户端静音。

名称	描述
mute	Yes：麦克风静音 No：取消静音
Return Value	0：方法调用成功 <0：方法调用失败

静音所有远端音频流 (muteAllRemoteAudioStreams)

```
(int)muteAllRemoteAudioStreams:(BOOL)muted;
```

将所有远端用户静音或取消静音。

名称	描述
muted	Yes：停止播放所有接收音频流 No：继续播放接收到的音频流
Return Value	0：方法调用成功 <0：方法调用失败

静音指定远端音频流 (muteRemoteAudioStream)

```
(int)muteRemoteAudioStream :(NSUInteger)uid
                    muted:(BOOL)muted;
```

将指定远端用户静音或取消静音。

注：该方法不影响音频数据流的接收，只是不播放音频流。

名称	描述
uid	指定用户
muted	Yes：停止播放指定用户音频流 No：继续播放指定用户音频流
Return Value	0：方法调用成功 <0：方法调用失败

扬声器通话 (setEnableSpeakerphone)

```
(int)setEnableSpeakerphone:(BOOL)enableSpeaker;
```

启用扬声器。

注：由于 SDK 调用 `setCategory AVAudioSessionCategoryPlayAndRecord withOptions` 设置听筒/扬声器状态，正在播放的声音会被打断。

名称	描述
enableSpeaker	Yes：切换到扬声器 No：切换到听筒
Return Value	0：方法调用成功 <0：方法调用失败

是否是扬声器状态 (isSpeakerphoneEnabled)

```
(BOOL)isSpeakerphoneEnabled;
```

当前是否是扬声器状态。

名称	描述

Return Value	Yes：是 No：否

启用说话者音量提示 (enableAudioVolumeIndication)

```
(int)enableAudioVolumeIndication:(NSInteger)interval
                          smooth:(NSInteger)smooth;
```

该方法允许 SDK 定期向应用程序反馈当前谁在说话以及说话者的音量。

名称	描述
interval	指定音量提示的时间间隔。 <=0 ：禁用音量提示功能 >0 ：提示间隔，单位为毫秒。建议设置到大于 200 毫秒。
smooth	平滑系数。默认可以设置为 3。
Return Value	0：方法调用成功 <0：方法调用失败

注册数据包观测器(registerAgoraPacketObserver)

```
int registerAgoraPacketObserver(void* nativeHandle,
agora::rtc::IPacketObserver* observer)
```

该方法注册数据包观测器(Packet Observer)。在 Agora SDK 发送/接收（语音、视频）网络包时，会回调 IPacketObserver 定义的接口，应用程序可用此接口对数据做处理，比如加解密。本 API 能直接与 mm 文件一起在 Objective-C 使用。

注：

- 确保在使用本方法前，应用程序已初始化 AgoraRtcEngineKit 实例。
- 处理后发送到网络的包大小不应超过 1200 字节，否则有可能发送失败。

名称	描述
nativeHandle	通过调用 API - (void*)getNativeHandle;获取。
observer	发送/接收封包的回调接口。
Return Value	0：方法调用成功 <0：方法调用失败

启用内置的加密方案(setEncryptionSecret)

```
(int)setEncryptionSecret:(NSString*)secret;
```

在加入频道之前，您的应用程序需调用 setEncryptionSecret 指定 secret 来启用内置的 AES-128 加密功能，同一频道内的所有用户应设置相同的 secret。 当用户离开频道时，

该频道的 secret 会自动清除。如果未指定 secret 或将 secret 设置为空，则无法激活加密功能。

名称	描述
secret	用来启用内置加密方案的参数
Return Value	0：方法调用成功 <0：方法调用失败

设置视频属性 (setVideoProfile)

```
(int)setVideoProfile:(AgoraRtcVideoProfile)profile;
```

该方法设置视频编码属性（Profile）。每个 Profile 对应一套视频参数，如分辨率、帧率、码率等。

注：应在调用 `joinChannel` 进入频道前设置视频属性。当设备的摄像头不支持指定的分辨率时，SDK 会自动选择一个合适的摄像头分辨率，但是编码分辨率仍然用 setVideoProfile 指定的。

名称	描述
profile	视频 Profile。细节见下表 `AgoraRtcVideoProfile` 的定义。
Return Value	0：方法调用成功 <0：方法调用失败

视频属性（profile）定义

视频属性	枚举值	分辨率(宽 x 高)	帧率(fps)	码率(kbps)
120P	0	160x120	15	80
120P_2	1	120x160	15	80
120P_3	2	120x120	15	60
180P	10	320x180	15	160
180P_2	11	180x320	15	160
180P_3	12	180x180	15	120
240P	20	320x240	15	200
240P_2	21	240x320	15	200
240P_3	22	240x240	15	160
360P	30	640x360	15	400

360P_2	31	360x640	15	400
360P_3	32	360x360	15	300
360P_4	33	640x360	30	800
360P_5	34	360x640	30	800
360P_6	35	360x360	30	600
480P	40	640x480	15	500
480P_2	41	480x640	15	500
480P_3	42	480x480	15	400
480P_4	43	640x480	30	1000
480P_5	44	480x640	30	1000
480P_6	45	480x480	30	800
480P_7	46	640x480	15	1000
720P	50	1280x720 *	15	1000
720P_2	51	720x1080 *	15	1000
720P_3	52	1280x720 *	30	2000
720P_4	53	720x1280 *	30	2000

* 视频能否达到 720P 的分辨率取决于设备的性能，在性能配备较低的设备上有可能无法实现。如果采用 720P 分辨率而设备性能跟不上，则有可能出现帧率过低的情况。Agora.io 将继续在后续版本中为较低端设备进行视频优化。如设备有特别需求，请联系 support@agora.io。

设置本地视频视图 (setupLocalVideo)

```
(int)setupLocalVideo:(AgoraRtcVideoCanvas*)local;
```

该方法设置本地视频显示信息。应用程序通过调用此接口绑定本地视频流的显示视窗(view)，并设置视频显示模式。在应用程序开发中， 通常在初始化后调用该方法进行本地视频设置，然后再加入频道。退出频道后，绑定仍然有效，如果需要解除绑定，可以指定空(NULL)View 调用 setupLocalVideo。

名称	描述
local	设置视频属性。AgoraRTCVideoCanvas 类型。 • view：视频显示视窗。SDK 不维护 view 的生命周期，应用程序应保证 view 在通话中是有效的，否则可能会导致 SDK 崩溃。view 可以在

	leaveChannel 调用返回后安全销毁。 • renderMode：视频显示模式。 AgoraRtc_Render_Hidden(1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。 AgoraRtc_Render_Fit(2)：如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗。 AgoraRtc_Render_Adaptive(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 AgoraRtc_Render_Hidden；如果对方和自己一个竖屏一个横屏，则使用 AgoraRtc_Render_Fit。 • uid：本地用户 ID，与 joinchannel 方法中的 uid 保持一致。
Return Value	0：方法调用成功 <0：方法调用失败

设置远端视频显示视图 (setupRemoteVideo)

```
(int)setupRemoteVideo:(AgoraRtcVideoCanvas*)remote;
```

该方法绑定远程用户和显示视图，即设定 uid 指定的用户用哪个视图显示。调用该接口时需要指定远程视频的 uid，一般可以在进频道前提前设置好，如果应用程序不能事先知道对方的 uid，可以在应用程序收到 didJoinedOfUid 事件时设置。如果启用了视频录制功能，视频录制服务会做为一个哑客户端加入频道，因此其他客户端也会收到它的 didJoinedOfUid 事件，APP 不应给它绑定视图（因为它不会发送视频流），如果 APP 不能识别哑客户端，可以在 firstRemoteVideoDecodedOfUid 事件触发时再绑定视图。

解除某个用户的绑定视图可以把 view 设置为空。

退出频道后，SDK 会把远程用户的绑定关系清除掉。

名称	描述
remote	设置视频属性。AgoraRTCVideoCanvas 类型。 • view：视频显示视窗。SDK 不维护 view 的生命周期，应用程序应保证 view 在通话中是有效的，否则可能会导致 SDK 崩溃。view 可以在 leaveChannel 调用返回后安全销毁。 • renderMode：视频显示模式。 AgoraRtc_Render_Hidden(1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗

	的比例进行周边裁剪或图像拉伸后填满视窗。 AgoraRtc_Render_Fit(2)：如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗。 AgoraRtc_Render_Adaptive(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 AgoraRtc_Render_Hidden；如果对方和自己一个竖屏一个横屏，则使用 AgoraRtc_Render_Fit。 • uid：用户 ID，指定远程视频来自哪个用户。
Return Value	0：方法调用成功 <0：方法调用失败

设置本地视频显示模式 (setLocalRenderMode)

```
(int)setLocalRenderMode:(AgoraRtcRenderMode) mode;
```

该方法设置本地视频显示模式。应用程序可以多次调用此方法更改显示模式。

名称	描述
mode	设置视频显示模式。AgoraRtcRenderMode 类型。 • AgoraRtc_Render_Hidden(1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。 • AgoraRtc_Render_Fit(2)：如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗。 • AgoraRtc_Render_Adaptive(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 AgoraRtc_Render_Hidden；如果对方和自己一个竖屏一个横屏，则使用 AgoraRtc_Render_Fit。
Return Value	0：方法调用成功 <0：方法调用失败

设置远端视频显示模式 (setRemoteRenderMode)

```
(int)setRemoteRenderMode: (NSUInteger)uid
mode:(AgoraRtcRenderMode) mode;
```

该方法设置远端视频显示模式。应用程序可以多次调用此方法更改显示模式。

名称	描述
uid	用户 ID，指定远程视频来自哪个用户。
mode	设置视频显示模式。AgoraRtcRenderMode 类型。 • AgoraRtc_Render_Hidden(1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。 • AgoraRtc_Render_Fit(2)：如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗. • AgoraRtc_Render_Adaptive(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 AgoraRtc_Render_Hidden；如果对方和自己一个竖屏一个横屏，则使用 AgoraRtc_Render_Fit。
Return Value	0：方法调用成功 <0：方法调用失败

切换前置/后置摄像头 (switchCamera)

```
(int)switchCamera;
```

该方法用于在前置/后置摄像头间切换。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

暂停发送本地视频流 (muteLocalVideoStream)

```
(int)muteLocalVideoStream:(BOOL)muted;
```

暂停/恢复发送本地视频流。该方法用于允许/禁止往网络发送本地视频流。

注：该方法不影响本地视频流获取，没有禁用摄像头。

名称	描述
mute	Yes: 暂停发送本地视频流 No: 恢复发送本地视频流
Return Value	0：方法调用成功 <0：方法调用失败

暂停所有远端视频流 (muteAllRemoteVideoStreams)

```
(int)muteAllRemoteVideoStreams:(BOOL)muted;
```

暂停/恢复所有人视频流。本方法用于允许/禁止播放所有人的视频流。

注：该方法不影响视频数据流的接收，只是不播放视频流。

名称	描述
mute	Yes: 停止播放接收到的所有视频流 No: 允许播放接收到的所有视频流
Return Value	0：方法调用成功 <0：方法调用失败

暂停指定远端视频流 (muteRemoteVideoStream)

```
(int)muteRemoteVideoStream :(NSUInteger)uid
                     muted:(BOOL)muted;
```

暂停/恢复指定用户视频流。本方法用于允许/禁止播放指定用户的视频流。

注：该方法不影响视频数据流的接收，只是不播放视频流。

名称	描述
uid	用户 ID
mute	Yes: 停止播放接收到的视频流 No: 允许播放接收到的视频流
Return Value	0：方法调用成功 <0：方法调用失败

开始客户端录音 (startAudioRecording)

```
(int)startAudioRecording:(NSString*)filePath;
```

开始录音。SDK 支持在通话中进行录音，录音文件的格式为 wav。应用程序必须保证指定的目录存在而且可写。该接口需要在 `joinChannel` 之后调用；在 `leaveChannel` 时如果还在录音，会自动停止。

名称	描述
filePath	存储录音文件的文件路径
Return Value	0：方法调用成功 <0：方法调用失败

停止客户端录音 (stopAudioRecording)

```
(int)stopAudioRecording;
```

停止录音。该接口需要在 `leaveChannel` 之前调用，如果没有调用，在 `leaveChannel` 时会自动停止。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

开始客户端本地混音(startAudioMixing)

```
(int) startAudioMixing: (NSString*) filePath
                loopback:(BOOL) loopback
                replace: (BOOL) replace
                 cycle: (NSInteger) cycle
```

指定本地音频文件来和麦克风采集的音频流进行混音和替换(用音频文件替换麦克风采集的音频流)，可以通过参数选择对方是否能听到本地播放的音频和指定循环播放的次数。

名称	描述
filePath	指定需要混音的本地音频文件名和文件路径。 支持以下音频格式： • mp3 • aac • m4a • 3gp • wav
loopback	True: 只有本地可以听到混音或替换后的音频流 False: 本地和对方都可以听到混音或替换后的音频流
replace	True: 音频文件内容将会替换本地录音的音频流 False: 音频文件内容将会和麦克风采集的音频流进行混音
cycle	指定音频文件循环播放的次数。 • 0：不循环 • 正整数：循环的次数 • -1：无限循环
Return Value	0：方法调用成功 <0：方法调用失败

停止客户端本地混音(stopAudioMixing)

```
(int)stopAudioMixing
```

停止本地音频文件混音。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

设置日志文件 (setLogFile)

```
(int)setLogFile:(NSString*)filePath;
```

设置 SDK 的输出 log 文件。SDK 运行时产生的所有 log 将写入该文件。应用程序必须保证指定的目录存在而且可写。

名称	描述
filePath	log 文件的全路径名
Return Value	0：方法调用成功 <0：方法调用失败

设置日志文件过滤器 (setLogFilter)

```
(int)setLogFilter:(NSUInteger)filter;
```

设置 SDK 的输出日志过滤器。不同的过滤器可以用或组合。

名称	描述
filter	过滤器： • 1: INFO • 2: WARNING • 4: ERROR • 8: FATAL • 0x800: DEBUG
Return Value	0：方法调用成功 <0：方法调用失败

启动服务端录音服务 (startRecordingService)

```
(int)startRecordingService: (NSString*)key
```

启动服务端录音功能。启用该服务需要正确的动态 Key。由于该调用是异步操作，调用后的执行结果（成功或失败）在 didApiCallExecute 回调中返回。

名称	描述
key	动态 Key
Return Value	0：方法调用成功 <0：方法调用失败

停止服务端录音服务 (stopRecordingService)

```
(int)stopRecordingService: (NSString*)key
```

停止服务端录音功能。启用该服务需要正确的动态 Key。由于该调用是异步操作，调用后的执行结果（成功或失败）在 `didApiCallExecute` 回调中返回。

名称	描述
key	动态 Key
Return Value	0：方法调用成功 <0：方法调用失败

服务端录音服务状态 (refreshRecordingServiceStatus)

```
(int)refreshRecordingServiceStatus
```

更新服务端录音状态。由于该调用是异步操作，调用成功后查询状态在 `didRefreshRecordingServiceStatus` 回调中返回。如果调用执行失败，结果在 `didApiCallExecute` 回调中返回。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

销毁引擎实例 (destroy)

```
public void destroy()
```

该方法释放 Agora SDK 使用的所有资源。有些应用程序只在用户需要时才进行音频或视频通信，不需要时则将资源释放出来用于其他操作，该方法对这类程序可能比较有用。一旦调用了 destroy(), 用户将无法再使用和回调该 SDK 内的其它方法。如需再次使用通信功能，必须重新初始化 sharedEngineWithVendorKey 来新建一个 AgoraRtcEngineKit 实例（instance）。

代理方法（AgoraRtcEngineDelegate）

SDK 会通过 Delegate 向应用程序上报一些运行时事件。从 1.1 版本开始，SDK 使用 Delegate 代替原有的部分 Block 回调。原有的 Block 回调被标为废弃，目前仍然可以使用，但是建议用相应的 Delegate 方法代替。如果同一个回调 Block 和 Delegate 方法都有定义，则 SDK 只回调 Block 方法。

发生警告回调 (didOccurWarning)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didOccurWarning:(AgoraRtcErrorCode)warningCode
```

该回调方法表示 SDK 运行时出现了（网络或媒体相关的）警告。通常情况下，SDK 上报的警告信息应用程序可以忽略，SDK 会自动恢复。比如和服务器失去连接时，SDK 可能会上报 `AgoraRtc_Error_OpenChannelTimeout(106)` 警告，同时自动尝试重连。

名称	描述
warningCode	警告代码

发生错误回调 (didOccurError)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didOccurError:(AgoraRtcErrorCode)errorCode
```

该回调方法表示 SDK 运行时出现了（网络或媒体相关的）错误。通常情况下，SDK 上报的错误意味着 SDK 无法自动恢复，需要应用程序干预或提示用户。比如启动通话失败时，SDK 会上报 AgoraRtc_Error_StartCall(1002)错误。应用程序可以提示用户启动通话失败，并调用 leaveChannel 退出频道。

名称	描述
Engine	AgoraRtcEngineKit 对象
errorCode	错误代码

加入频道成功回调 (didJoinChannel)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didJoinChannel:(NSString*)channel withUid:(NSUInteger)uid
elapsed:(NSInteger) elapsed
```

同 joinChannelByKey API 的 joinSuccessBlock。该回调方法表示该客户端成功加入了指定的频道。

名称	描述
channel	频道名
uid	用户 ID。如果 joinChannelByKey 中指定了 uid，则此处返回该 ID；否则使用 Agora 服务器自动分配的 ID。
elapsed	从 joinChannel 开始到该事件产生的延迟（毫秒）

重新加入频道回调 (didRejoinChannel)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didRejoinChannel:(NSString*)channel withUid:(NSUInteger)uid
elapsed:(NSInteger) elapsed
```

有时候由于网络原因，客户端可能会和服务器失去连接，SDK 会进行自动重连，自动重连成功后触发此回调方法，提示有用户重新加入了频道，且频道 ID 和用户 ID 均已分配。

名称	描述
channel	频道 ID
uid	用户 ID
elapsed	加入延迟（毫秒）

音量提示回调 (reportAudioVolumeIndicationOfSpeakers)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
reportAudioVolumeIndicationOfSpeakers:(NSArray*)speakers
totalVolume:(NSInteger)totalVolume
```

同 audioVolumeIndicationBlock。提示谁在说话及其音量，默认禁用。可通过 enableAudioVolumeIndication 方法设置。

名称	描述
speakers	说话者（数组）。每个 speaker()： • uid：说话者的用户 ID • volume：说话者的音量（0~255）
totalVolume	（混音后的）总音量（0~255）

本地首帧视频显示回调 (firstLocalVideoFrameWithSize)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
firstLocalVideoFrameWithSize:(CGSize)size
elapsed:(NSInteger)elapsed
```

同 firstLocalVideoFrameBlock。提示第一帧本地视频画面已经显示在屏幕上。

名称	描述
size	视频流尺寸（宽度和高度）
elapsed	加入频道开始到该回调触发的延迟（毫秒）

远端首帧视频接收解码回调 (firstRemoteVideoDecodedOfUid)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
firstRemoteVideoDecodedOfUid:(NSUInteger)uid size:(CGSize)size
elapsed:(NSInteger)elapsed
```

同 firstRemoteVideoDecodedBlock。提示已收到第一帧远程视频流并解码。

名称	描述
uid	用户 ID，指定是哪个用户的视频流
size	视频流尺寸（宽度和高度）
elapsed	加入频道开始到该回调触发的延迟（毫秒）

远端首帧视频显示回调 (firstRemoteVideoFrameOfUid)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
firstRemoteVideoFrameOfUid:(NSUInteger)uid size:(CGSize)size
elapsed:(NSInteger)elapsed
```

同 firstRemoteVideoFrameBlock。提示第一帧远端视频画面已经显示在屏幕上。

名称	描述

uid	用户 ID，指定是哪个用户的视频流
size	视频流尺寸（宽度和高度）
elapsed	加入频道开始到该回调触发的延迟（毫秒）

用户加入回调 (didJoinedOfUid)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didJoinedOfUid:(NSUInteger)uid elapsed:(NSInteger)elapsed
```

同 userJoinedBlock。提示有用户加入了频道。如果该客户端加入频道时已经有人在频道中，SDK 也会向应用程序上报这些已在频道中的用户。

名称	描述
uid	用户 ID。如果 joinChannel 中指定了 uid，则此处返回该 ID；否则使用 Agora 服务器自动分配的 ID。
elapsed	从 joinChannel 开始到该事件产生的延迟（毫秒）

用户离线回调 (didOfflineOfUid)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didOfflineOfUid:( NSUInteger )uid
reason:(AgoraRtcUserOfflineReason)reason
```

同 userOfflineBlock。提示有用户离开了频道（或掉线）。

注：SDK 判断用户离开频道（或掉线）的依据是超时：在一定时间内（15 秒）没有收到对方的任何数据包，判定为对方掉线。在网络较差的情况下，可能会有误报。建议可靠的掉线检测应该由信令来做。

名称	描述
uid	用户 ID。
reason	离线原因： • AgoraRtc_UserOffline_Quit：用户主动离开 • ßAgoraRtc_UserOffline_Dropped：因过长时间收不到对方数据包，超时掉线。注意：由于 SDK 使用的是不可靠通道，也有可能对方主动离开本方没收到对方离开消息而误判为超时掉线。

用户音频静音回调 (didAudioMuted)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didAudioMuted:(BOOL)muted byUid:(NSUInteger)uid
```

同 userMuteAudioBlock。提示有用户用户将通话静音/取消静音。

名称	描述
uid	用户 ID
muted	Yes：静音

	No：取消静音

用户停止/重新发送视频回调 (didVideoMuted)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didVideoMuted:(BOOL)muted byUid:(NSUInteger)uid
```

同 userMuteVideoBlock。提示有其他用户暂停发送/恢复发送其视频流。

名称	描述
uid	用户 ID
muted	Yes：该用户已暂停发送其视频流 No：该用户已恢复发送其视频流

用户启用/关闭视频回调 (didVideoEnabled)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didVideoEnabled:(BOOL)enabled byUid:(NSUInteger)uid
```

提示有其他用户启用/关闭了视频功能。关闭视频功能是指该用户只能进行语音通话，不能显示、发送自己的视频，也不能接收、显示别人的视频。

名称	描述
uid	用户 ID
enabled	Yes：该用户已启用了视频功能 No：该用户已关闭了视频功能

本地视频统计回调 (localVideoStats)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
localVideoStats:(AgoraRtcLocalVideoStats*)stats
```

同 `localVideoStatBlock`。报告更新本地视频统计信息，该回调方法每两秒触发一次。`AgoraRtcLocalVideoStats` 定义如下：

名称	描述
sentBytes	（上次统计后）发送的字节数
sentFrames	（上次统计后）发送的帧数

远端视频统计回调 (remoteVideoStats)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
remoteVideoStats:(AgoraRtcRemoteVideoStats*)stats
```

同 `remoteVideoStatBlock`。报告更新远端视频统计信息，该回调方法每两秒触发一次。

名称	描述
uid	用户 ID，指定是哪个用户的视频流

delay	延时(毫秒)
width	视频流宽（像素）
height	视频流高（像素）
receivedBitrate	（上次统计后）接收到的码率(kbps)
receivedFrameRate	（上次统计后）接收帧率(fps)

摄像头启用回调 (rtcEngineCameraDidReady)

```
(void)rtcEngineCameraDidReady:(AgoraRtcEngineKit *)engine
```

同 cameraReadyBlock。提示已成功打开摄像头，可以开始捕获视频。

视频功能停止回调(rtcEngineVideoDidStop)

```
(void)rtcEngineVideoDidStop:(AgoraRtcEngineKit *)engine
```

提示视频功能已停止。应用程序如需在停止视频后对 view 做其他处理（比如显示其他画面），可以在这个回调中进行。

网络连接中断回调 (rtcEngineConnectionDidInterrupted)

```
(void)rtcEngineConnectionDidInterrupted:(AgoraRtcEngineKit
*)engine
```

在 SDK 和服务器失去了网络连接时，触发该回调。失去连接后，除非 APP 主动调用 leaveChannel，SDK 会一直自动重连。

网络连接丢失回调 (rtcEngineConnectionDidLost)

```
(void)rtcEngineConnectionDidLost:(AgoraRtcEngineKit *)engine
```

在 SDK 和服务器失去了网络连接后，会触发 rtcEngineConnectionDidInterrupted 回调，并自动重连。在一定时间内（默认 10 秒）如果没有重连成功，触发 rtcEngineConnectionDidLost 回调。除非 APP 主动调用 leaveChannel，SDK 仍然会自动重连。

Rtc Engine 统计数据回调 (reportRtcStats)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
reportRtcStats:(AgoraRtcStats*)stats
```

同 rtcStatsBlock。该回调定期上报 Rtc Engine 的运行时的状态，每两秒触发一次。

名称	描述
stats	见下表

AgoraRtcStats	描述

duration	通话时长，累计值
txBytes	发送字节数，累计值
rxBytes	接收字节数，累计值

语音质量回调 (audioQualityOfUid)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
audioQualityOfUid:(NSUInteger)uid
quality:(AgoraRtcQuality)quality delay:(NSUInteger)delay
lost:(NSUInteger)lost
```

同 audioQualityBlock。在通话中，该回调方法每两秒触发一次，报告当前通话的（嘴到耳）音频质量。

名称	描述
audioQualityBlock	
uid	用户 ID。每次回调上报一个用户当前的语音质量（不包含自己），不同用户在不同的回调中上报。
quality	声音质量评分： AgoraRtc_Quality_Excellent(1) AgoraRtc_Quality_Good(2) AgoraRtc_Quality_Poor(3) AgoraRtc_Quality_Bad(4) AgoraRtc_Quality_VBad(5) AgoraRtc_Quality_Down(6)
delay	延迟（毫秒）
lost	丢包率（百分比）

网络质量回调 (networkQuality)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
networkQuality:(AgoraRtcQuality)quality
```

同 networkQualityBlock。报告网络质量，该回调方法每两秒触发一次。

名称	描述
quality	网络质量评分： AgoraRtc_Quality_Unknown = 0 AgoraRtc_Quality_Excellent = 1 AgoraRtc_Quality_Good = 2 AgoraRtc_Quality_Poor = 3 AgoraRtc_Quality_Bad = 4 AgoraRtc_Quality_VBad = 5 AgoraRtc_Quality_Down = 6

更新录音服务状态回调 (didRefreshRecordingServiceStatus)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didRefreshRecordingServiceStatus:(NSInteger)status;
```

该回调返回 refreshRecordingServiceStatus 执行成功后的状态码。

名称	描述
Status Value	0：录制停止 1：正在录制

录制开始/停止/状态查询回调(didApiCallExecute)

```
(void)rtcEngine:(AgoraRtcEngineKit *)engine
didApiCallExecute:(NSString*)api error:(NSInteger)error;
```

表示 API StartRecordingService 和 stopRecordingService 执行**成功或失败**的回调和状态码，以及 API refreshRecordingServiceStatus 执行**失败**的回调和状态码。

方法	API 字符串参数	状态码
StartRecordingService	rtc.api.start_recording_service	0: AgoraRtc_Error_NoError(成功) 1: AgoraRtc_Error_Failed (执行请求失败) 2: AgoraRtc_Error_InvalidArgument（参数无效） 3: ERR_NOT_READY（未就绪） 7: AgoraRtc_Error_NotInitialized （录音服务器 IP 未初始化） 10: AgoraRtc_Error_Timedout（操作超时）
stopRecordingService	rtc.api.stop_recording_service	
refreshRecordingServiceStatus	rtc.api.query_recording_service_status	状态码可能为上述 1，2，3，7 或 10. 描述同上。

回调方法

SDK 会通过 Block 给应用程序上报一些运行时事件，以下为 SDK 中的回调方法说明。

用户加入回调 (userJoinedBlock)

```
(void)userJoinedBlock:(void(^)(NSUInteger uid, NSInteger
elapsed))userJoinedBlock;
```

提示有用户加入了频道。如果该客户端加入频道时已经有人在频道中，SDK 也会向应用程序上报这些已在频道中的用户。

名称	描述
uid	用户 ID。如果 `joinChannel` 中指定了 uid，则此处返回该 ID；否则使用 Agora 服务器自动分配的 ID。
elapsed	从 `joinChannel` 开始到该事件产生的延迟（毫秒）

用户离线回调 (userOfflineBlock)

```
(void)userOfflineBlock:(void(^)(NSUInteger uid))userOfflineBlock;
```

提示有用户离开了频道（或掉线）。注：SDK 判断用户离开频道（或掉线）的依据是超时：在一定时间内（15 秒）没有收到对方的任何数据包，判定为对方掉线。在网络较差的情况下，可能会有误报。建议可靠的掉线检测应该由信令来做。

名称	描述
uid	用户 ID

重新加入频道回调 (rejoinChannelSuccessBlock)

```
(void)rejoinChannelSuccessBlock:(void(^)(NSString* channel,
NSUInteger uid, NSInteger elapsed))rejoinChannelSuccessBlock;
```

有时候由于网络原因，客户端可能会和服务器失去连接，SDK 会进行自动重连，自动重连成功后触发此回调方法，提示有用户重新加入了频道，且频道 ID 和用户 ID 均已分配。

名称	描述
channel	频道 ID
uid	用户 ID
elapsed	加入延迟（毫秒）

离开频道回调 (leaveChannelBlock)

```
(void)leaveChannelBlock:(void(^)(AgoraRtcStats*
stat))leaveChannelBlock;
```

显示用户离开了频道，提供会话数据信息，包括通话时长和 tx/rx 字节数。

名称	描述
stat	见下表

AgoraRtcStats	描述
duration	通话时长，累计值
txBytes	发送字节数，累计值
rxBytes	接收字节数，累计值

Rtc Engine 统计数据回调 (rtcStatsBlock)

```
(void)rtcStatsBlock:(void(^)(AgoraRtcStats* stat))rtcStatsBlock;
```

该回调定期上报 Rtc Engine 的运行时的状态，每两秒触发一次。

名称	描述
stat	见上表

音量提示回调 (audioVolumeIndicationBlock)

```
(void)audioVolumeIndicationBlock:(void(^)(NSArray *speakers,
NSInteger totalVolume))audioVolumeIndicationBlock;
```

提示谁在说话及其音量，默认禁用。可通过 `enableAudioVolumeIndication` 方法设置。

名称	描述
speakers	说话者（数组）。每个 speaker()： • uid：说话者的用户 ID • volume：说话者的音量（0~255）
totalVolume	（混音后的）总音量（0~255）

本地首帧视频显示回调 (firstLocalVideoFrameBlock)

```
(void)firstLocalVideoFrameBlock:(void(^)(NSInteger width,
NSInteger height, NSInteger elapsed))firstLocalVideoFrameBlock;
```

提示第一帧本地视频画面已经显示在屏幕上。

名称	描述
width	视频流宽(像素)
height	视频流高(像素)
elapsed	加入频道开始到该回调触发的延迟（毫秒）

远端首帧视频显示回调 (firstRemoteVideoFrameBlock)

```
(void)firstRemoteVideoFrameBlock:(void(^)(NSUInteger uid,
NSInteger width, NSInteger height, NSInteger
elapsed))firstRemoteVideoFrameBlock;
```

提示第一帧远端视频画面已经显示在屏幕上。

名称	描述
uid	用户 ID，指定是哪个用户的视频流
width	视频流宽(像素)
height	视频流高(像素)
elapsed	加入频道开始到该回调触发的延迟（毫秒）

远端首帧视频接收解码回调 (firstRemoteVideoDecodedBlock)

```
(void)firstRemoteVideoDecodedBlock:(void(^)(NSUInteger uid,
NSInteger width, NSInteger height, NSInteger
elapsed))firstRemoteVideoDecodedBlock;
```

提示已收到第一帧远程视频流并解码。

名称	描述
uid	用户 ID，指定是哪个用户的视频流
width	视频流宽（像素）
height	视频流高（像素）
elapsed	加入频道开始到该回调触发的延迟（毫秒）

用户音频静音回调 (userMuteAudioBlock)

```
(void)userMuteAudioBlock:(void(^)(NSUInteger uid, bool
muted))userMuteAudioBlock;
```

提示有用户用户将通话静音/取消静音。

名称	描述
uid	用户 ID
muted	Yes：静音 No：取消静音

用户停止/重新发送视频回调 (userMuteVideoBlock)

```
(void)userMuteVideoBlock:(void(^)(NSUInteger uid, bool
muted))userMuteVideoBlock;
```

提示有其他用户暂停发送/恢复发送其视频流。

名称	描述
uid	用户 ID
muted	Yes：该用户已暂停发送其视频流 No：该用户已恢复发送其视频流

本地视频统计回调 (localVideoStatBlock)

```
(void)localVideoStatBlock:(void(^)(NSInteger sentBytes,
NSInteger sentFrames))localVideoStatBlock;
```

报告更新本地视频统计信息，该回调方法每两秒触发一次。

名称	描述
sentBytes	（上次统计后）发送的字节数
sentFrames	（上次统计后）发送的帧数

远端视频统计回调 (remoteVideoStatBlock

```
(void)remoteVideoStatBlock:(void(^)(NSUInteger uid, NSInteger
frameCount, NSInteger delay, NSInteger
receivedBytes))remoteVideoStatBlock;
```

报告更新远端视频统计信息，该回调方法每两秒触发一次。

名称	描述
uid	用户 ID，指定是哪个用户的视频流
frameCount	（上次统计后）接收的总帧数
delay	延时(毫秒)
receivedBytes	（上次统计后）接收的总字节数

摄像头启用回调 (cameraReadyBlock)

```
(void)cameraReadyBlock:(void(^)())cameraReadyBlock;
```

提示已成功打开摄像头，可以开始捕获视频。

语音质量回调 (audioQualityBlock)

```
(void)audioQualityBlock:(void(^)(NSUInteger uid, AgoraRtcQuality
quality,  NSUInteger delay, NSUInteger lost))audioQualityBlock;
```

在通话中，该回调方法每两秒触发一次，报告当前通话的（嘴到耳）音频质量。

名称	描述
audioQualityBlock	
uid	用户 ID。每次回调上报一个用户当前的语音质量（不包含自己），不同用户在不同的回调中上报。
quality	声音质量评分： AgoraRtc_Quality_Excellent(1) AgoraRtc_Quality_Good(2) AgoraRtc_Quality_Poor(3) AgoraRtc_Quality_Bad(4) AgoraRtc_Quality_VBad(5) AgoraRtc_Quality_Down(6)
delay	延迟（毫秒）
lost	丢包率（百分比）

网络质量回调 (networkQualityBlock)

```
(void)networkQualityBlock:(void(^)(AgoraRtcQuality
quality))networkQualityBlock;
```

报告网络质量，该回调方法每两秒触发一次。

名称	描述

quality	网络质量评分： AgoraRtc_Quality_Unknown = 0 AgoraRtc_Quality_Excellent = 1 AgoraRtc_Quality_Good = 2 AgoraRtc_Quality_Poor = 3 AgoraRtc_Quality_Bad = 4 AgoraRtc_Quality_VBad = 5 AgoraRtc_Quality_Down = 6

网络连接丢失回调 (connectionLostBlock)

```
(void)connectionLostBlock:(void(^)())connectionLostBlock;
```

该回调方法表示 SDK 与服务器失去了网络连接。

附录

Agora SDK 在调用 API 或运行时会返回错误或警告代码：

- **错误代码**意味着 SDK 遭遇不可恢复的错误，需要应用程序干预，例如打开摄像头失败会返回错误，应用程序需要提示用户不能使用摄像头。
- **警告代码**意味着 SDK 遇到问题，但有可能恢复，警告代码仅起**告知**作用，一般情况下应用程序可以忽略警告代码。

错误代码(Error Code)

错误代码	值	描述
ERR_OK	0	正常情况的返回值，意味着没有错误。
ERR_FAILED	1	一般性的错误(没有明确归类的错误原因)。
ERR_INVALID_ARGUMENT	2	API 调用了无效的参数。例如指定的频道名含有非法字符。
ERR_NOT_READY	3	SDK 的模块没有准备好，例如某个 API 调用依赖于某个模块，但该模块尚未准备提供服务。
ERR_NOT_SUPPORTED	4	SDK 不支持该功能。
ERR_REFUSED	5	调用被拒绝。仅供 SDK 内部使用，不通过 API 或者回调事件返回给应用程序。
ERR_BUFFER_TOO_SMALL	6	传入的缓冲区大小不足以存放返回的数据。
ERR_NOT_INITIALIZED	7	SDK 尚未初始化，就调用其 API。
ERR_INVALID_VIEW	8	指定的 view 无效，使用视频功能时需要指定 view，如果 view 尚未指定，则返回该错误。
ERR_NO_PERMISSION	9	没有操作权限。仅供 SDK 内部使用，不通过 API 或者回调事件返回给应用程序。
ERR_TIMEDOUT	10	API 调用超时。有些 API 调用需要 SDK 返回结果，如果 SDK 处理时间过长，会出现此错误。
ERR_CANCELED	11	请求被取消。仅供 SDK 内部使用，不通过 API 或者回调事件返回给应用程序。
ERR_TOO_OFTEN	12	调用频率太高。仅供 SDK 内部使用，不通过

		API 或者回调事件返回给应用程序。
ERR_BIND_SOCKET	13	SDK 内部绑定到网络 socket 失败。仅供 SDK 内部使用，不通过 API 或者回调事件返回给应用程序。
ERR_NET_DOWN	14	网络不可用。仅供 SDK 内部使用，不通过 API 或者回调事件返回给应用程序。
ERR_NET_NOBUFS	15	没有网络缓冲区可用。仅供 SDK 内部使用，不通过 API 或者回调事件返回给应用程序。
ERR_INIT_VIDEO	16	初始化视频功能失败。
ERR_JOIN_CHANNEL_REJECTED	17	加入频道被拒绝。一般是因为用户已进入频道，再次调用加入频道的 API，例如 joinChannel，会返回此错误。
ERR_LEAVE_CHANNEL_REJECTED	18	离开频道失败。一般是因为用户已离开某频道，再次调用退出频道的 API，例如 leaveChannel，会返回此错误。
ERR_ALREADY_IN_USE	19	资源已被占用，不能重复使用
ERR_ALREADY_IN_USE	20	SDK 放弃请求，可能由于请求次数太多。仅供 SDK 内部使用，不通过 API 或者回调事件返回给应用程序。
ERR_INVALID_VENDOR_KEY	101	指定的 Vendor Key 无效。
ERR_INVALID_CHANNEL_NAME	102	指定的频道名无效。
ERR_DYNAMIC_KEY_TIMEOUT	109	当前使用的动态 key 过期，不再有效。
ERR_INVALID_DYNAMIC_KEY	110	指定的动态 key 无效。一般是因为 key 生成的不对。
ERR_LOAD_MEDIA_ENGINE	1001	加载媒体引擎失败。
ERR_START_CALL	1002	启动媒体引擎开始通话失败。
ERR_START_CAMERA	1003	启动摄像头失败。
ERR_START_VIDEO_RENDER	1004	启动视频渲染模块失败。
ERR_ADM_GENERAL_ERROR	1005	音频设备模块出现一般性错误（没有明显归类的错误）。
ERR_ADM_JAVA_RESOURCE	1006	音频设备模块： 使用 java 资源出现错误。
ERR_ADM_SAMPLE_RATE	1007	音频设备模块：

		设置的采样频率出现错误。
ERR_ADM_INIT_PLAYOUT	1008	音频设备模块： 初始化播放设备出现错误。
ERR_ADM_START_PLAYOUT	1009	音频设备模块： 启动播放设备出现错误。
ERR_ADM_STOP_PLAYOUT	1010	音频设备模块： 停止播放设备出现错误。
ERR_ADM_INIT_RECORDING	1011	音频设备模块： 初始化录音设备时出现错误。
ERR_ADM_START_RECORDING	1012	音频设备模块： 启动录音设备出现错误。
ERR_ADM_STOP_RECORDING	1013	音频设备模块： 停止录音设备出现错误。
ERR_ADM_RUNTIME_PLAYOUT_ERROR	1015	音频设备模块: 运行时播放出现错误。
ERR_ADM_RUNTIME_RECORDING_ERROR	1017	音频设备模块: 运行时录音错误。
ERR_ADM_RECORD_AUDIO_FAILED	1018	音频设备模块: 录音失败。
ERR_ADM_INIT_LOOPBACK	1022	音频设备模块: 初始化 loopback 设备错误。
ERR_ADM_START_LOOPBACK	1023	音频设备模块: 启动 loopback 设备错误。
ERR_VDM_CAMERA_NOT_AUTHORIZED	1501	视频设备模块: 摄像头没有使用权限。

警告代码(Warning Code)

警告代码	值	描述
WARN_PENDING	20	请求处于待定状态。一般是由于某个模块还没准

		备好，请求被延迟处理。
WARN_NO_AVAILABLE_CHANNEL	103	没有可用的频道资源。可能是因为服务端没法分配频道资源。
WARN_LOOKUP_CHANNEL_TIMEOUT	104	查找频道超时。在加入频道时 SDK 先要查找指定的频道，出现该警告一般是因为网络太差，连接不到服务器。
WARN_LOOKUP_CHANNEL_REJECTED	105	查找频道请求被服务器拒绝。服务器可能没有办法处理这个请求或请求是非法的。
WARN_OPEN_CHANNEL_TIMEOUT	106	打开频道超时。查找到指定频道后，SDK 接着打开该频道，超时一般是因为网络太差，连接不到服务器。
WARN_OPEN_CHANNEL_REJECTED	107	打开频道请求被服务器拒绝。服务器可能没有办法处理该请求或该请求是非法的。
WARN_ADM_RUNTIME_PLAYOUT_WARNING	1014	音频设备模块: 运行时播放设备出现警告。
WARN_ADM_RUNTIME_RECORDING_WARNING	1016	音频设备模块: 运行时录音设备出现警告。
WARN_ADM_RECORD_AUDIO_SILENCE	1019	音频设备模块: 没有采集到有效的声音数据。
WARN_ADM_PLAYOUT_MALFUNCTION	1020	音频设备模块： 播放设备出现故障。
WARN_ADM_RECORD_MALFUNCTION	1021	音频设备模块: 录制设备出现故障。
WARN_APM_HOWLING	1051	音频处理模块: 检测到啸叫。