



Agora Native SDK for Web

参考手册

1.7 版

support@agora.io

目录

简介	4
功能	4
用户须知	5
兼容性	5
浏览器支持	5
操作系统	5
所需文档	5
功能和局限性	5
入门指南	6
获取 SDK	6
设置密钥方案	6
密钥介绍	6
获取和使用 App ID	9
获取和使用 Channel Key	9
程序开发和部署	13
运行示例网页程序	14
客户端	14
服务器	14
加密数据	15
安装 AgoraWebAgent (Windows 或 Mac 平台)	15
Windows 平台	15
Mac 平台	17
卸载 AgoraWebAgent (Windows 或 Mac 平台)	20
Windows 平台	20
Mac 平台	20
Agora Native SDK for Web - API 参考	22
AgoraRTC 接口类	22
创建客户端对象(createRtcClient)	22
创建本地音视频流对象(createStream)	22
Client 接口类	23
初始化客户端对象(init)	23
加入 AgoraRTC 频道(join)	24
更新 Channel Key(renewChannelKey)	25
离开 AgoraRTC 频道(leave)	26
上传音视频流(publish)	26
取消上传音视频流 (unpublish)	27

接收远程音视频流(subscribe).....	27
取消接收远程音视频流 (unsubscribe).....	28
枚举平台设备 (getDevices)	28
启用内置的加密密码(setEncryptionSecret).....	29
设置内置的加密方案(setEncryptionMode)	29
选择设备 (selectDevice).....	30
开始录制 (startRecording)	30
停止录制(stopRecording)	31
查询录制状态(queryRecordingStatus)	32
获取桌面窗口列表(getWindows).....	32
开启屏幕共享(startScreenSharing)	32
指定共享窗口(setScreenSharingWindow)	33
停止屏幕共享(stopScreenSharing).....	33
清理资源(close)	33
远程音视频流已添加事件(stream-added)	34
远程音视频流已订阅事件(stream-subscribed)	34
远端用户已离开房间事件(peer-leave).....	34
远端用户视频发送事件(peer-mute-video).....	35
远端用户设置静音事件(peer-mute-audio)	35
网络连接中断或丢失事件(error).....	36
Stream 接口类.....	37
初始化本地音视频对象(init).....	37
关闭视频流(close).....	37
播放音/视频流(play).....	37
停止播放视频流(stop).....	38
启用音频流(enableAudio)	38
禁用音频流(disableAudio)	38
启用视频流(enableVideo)	38
禁用视频流(disableVideo)	38
获取用户 ID(getId)	38
设置视频编码属性(setVideoProfile)	39
附录	41
错误代码(Error Code)	41
警告代码(Warning Code).....	44

简介

Agora Native SDK for Web 提供音视频通话功能，且 Native SDK 在程序生成时直接与应用程序建立连接。

功能

Agora Native SDK for Web 包含 Agora JavaScript SDK 和一个应用程序 AgoraWebAgent。**Agora Native SDK for Web** 可实现以下功能：

- **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、选择摄像头等。
- **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 Recording Key 的用户。
- **数据加密：**对语音和视频数据进行加密。
- **屏幕共享：**将桌面或窗口共享给对方。启用屏幕共享功能后，对方屏幕上将显示您共享的桌面或窗口，而不是您摄像头拍摄的视频。
- **白板功能：**提供白板功能，供用户画图、注释、上传文件等功能。

Agora Native SDK for Web 提供 3 个 JavaScript 接口类实现以下功能：

- 顶层 **AgoraRTC** 对象用于创建相应的 **Client** 和 **Stream** 对象。
- 单个 **Client** 对象用于建立和管理通话。
- 多个 **Stream** 对象用于管理不同的音视频流。

详情参考章节 [Agora Native SDK for Web - API 参考](#)。

Agora SDK 在调用 API 或运行过程中，可能会返回错误或警告代码，详见[附录](#)。

用户须知

兼容性

Native SDK 1.3 版或更高版本

浏览器支持

不要使用暂未支持的浏览器。例如 Opera, Firefox 等浏览器暂不支持，避免产生不良后果。

平台	浏览器	要求
Windows	Edge	25.10586.0.0
Windows 10	Chrome*	51.0.2704.103 (64 位) ^注
Mac	Safari	9.1.1(11601.6.17)

^注:

目前在 Windows 10 上可使用 Chrome 浏览器，但仅支持分辨率 640x480p 30fps，后续将支持更高分辨率。

操作系统

操作系统	要求
Windows	Windows 7 或更高版本
Mac	OS X v10.10 或更高版本

所需文档

- 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。
- **Agora Whiteboard SDK 参考手册**：如需使用白板功能，请结合本参考手册一起使用。

所有软件包以及文档，均可通过以下网址下载：

<http://cn.agora.io/download/>

功能和局限性

- 推荐的最大视频分辨率为 720p 15fps 和 480p 30fps。
- 一台 PC 上只能运行一个实例。

入门指南

本章节介绍如何使用 Agora Native SDK for Web。

获取 SDK

请登录 <http://cn.agora.io/download/> 网站下载或联系 sales@agora.io 获取最新版的 **Agora Native SDK for Web**。

Agora Native SDK for Web 工具包内包含以下文件：

组件	描述
./agent ¹	包括以下安装程序： AgoraWebAgent.pkg AgoraWebAgentSetup.exe
./client	示例网页程序
./doc	包括以下参考文档(中英文): Agora Native SDK for Web Reference Manual v1_7_EN.pdf Agora Native SDK for Web Reference Manual v1_7_CHS.pdf
./lib	所需库
./server	Web 版服务器端示例代码

1. 完成部署以后，该安装程序还能从客户端系统提示上下载。更多信息，详见章节[安装 AgoraWebAgent \(Windows 或 Mac 平台\)](#)。

设置密钥方案

密钥介绍

静态密钥:

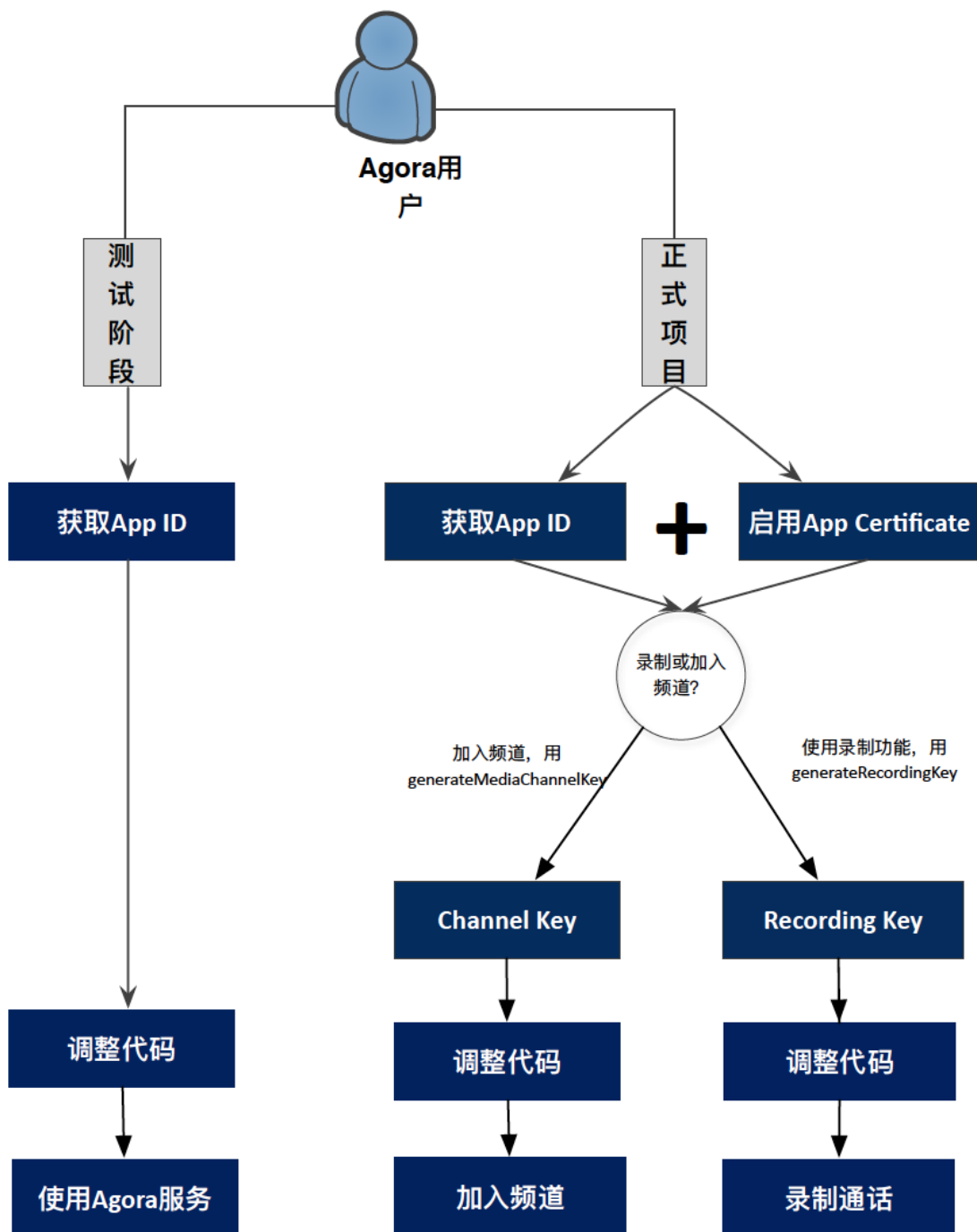
每个在 Agora 官网注册的账户均能创建多个项目，且每个项目均有唯一的 App ID。在 Agora Global Network 上的通信按照 App ID 隔离，持有不同 App ID 的用户即使加入的频道名称相同也会隔离，不会互相干扰。但 App ID 是一种静态密钥，如果有人非法获取了您的 App ID，他将可以在 Agora 提供的 SDK 中使用您的 App ID，如果他知道您的频道名字，甚至有可能干扰您正常的通话。

动态密钥:

Dynamic Keys 提供更为安全的用户身份验证方案，是动态密钥的统称，针对不同的服务，Dynamic Key 有不同的名称，Agora Native SDK 可能用到的 Dynamic Key 包括 Channel Key 和 Recording Key，分别用于加入频道和使用录制功能:

- **Channel Key:** 用户加入频道时，后台服务通过 Agora 提供的 App ID 和 App Certificate, 基于 HMAC 的安全算法生成一个新的 Channel Key 发送给客户端，客户端调用加入频道的接口函数使用此 Channel Key。Agora 的服务器通过 Channel Key 验证用户是否合法。该密钥由 generateMediaChannelKey 方法生成。
- **Recording Key:** 用户开始录制前，后台服务通过 Agora 提供的 App ID 和 App Certificate, 基于 HMAC 的安全算法生成一个新的 Recording Key 发送给客户端，客户端调用开始录制的接口函数使用此 Recording Key。Agora 的服务器通过 Recording Key 验证用户是否合法。该密钥由 generateRecordingKey 方法生成。

下图列出各类密钥的关系及使用场景:



注：

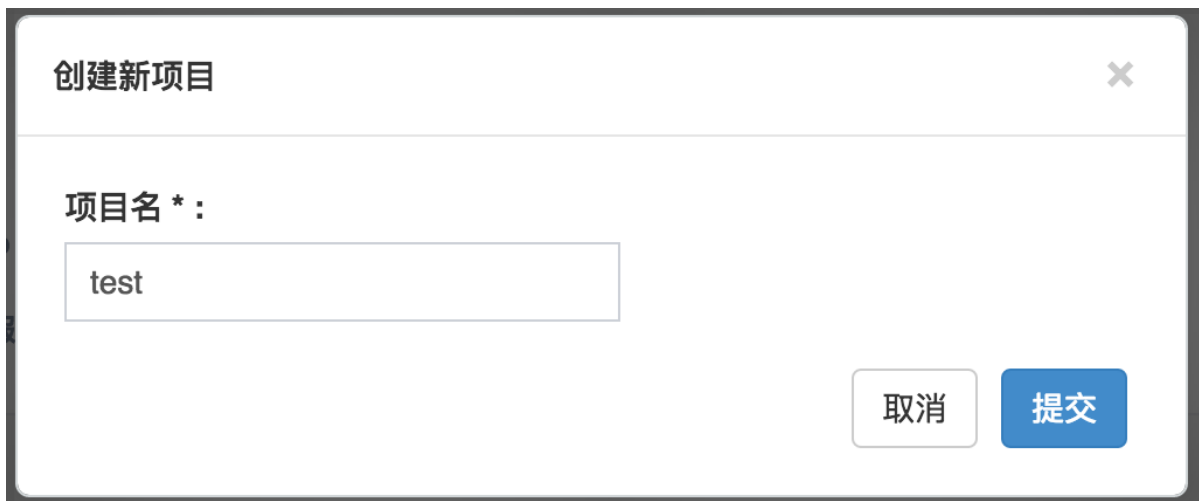
1. App ID适用于测试阶段或对安全性要求不高的场景。
2. 如需录制，必须使用Recording Key。
3. App Certificate仅用来生成Channel Key或Recording Key，不可单独使用。

获取和使用 App ID

获取 App ID

每个 Agora 账户下可创建多个项目，且每个项目有独立的 App ID。

1. 访问 <https://dashboard.agora.io/>，点击右上角的注册。
2. 在项目列表页面点击添加新项目。
3. 填写项目名。点击提交。



4. 在所创建的项目下查看 App ID。获取 App ID 后即可使用对应的 Agora.io 服务功能。

注：

关于如何运行示例程序（针对 App ID 和 Channel Key），请参考章节[运行示例网页程序](#)。

使用 App ID

使用每个项目下唯一的 App ID 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 App ID。
2. 在开发程序时，将 App ID 嵌入您的代码中以便调用 SDK。
3. 当调用 `init()` 初始化客户端对象时，将参数 `appId` 设为您的 App ID。
4. 当调用 `AgoraRTC.client.join()` 加入频道时，将参数 `key` 设为 NULL。

获取和使用 Channel Key

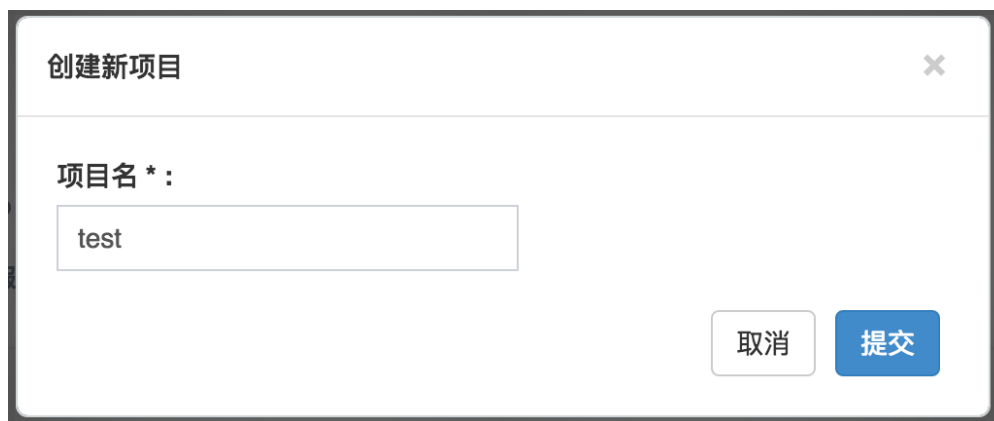
本章介绍如何获取和使用 Channel Key，Recording Key 的步骤和说明与此类似，如有需要，详见 *Agora Recording Server 用户指南*。

获取 App ID 和 App Certificate

每个 Agora 账户下可创建多个项目，且每个项目有独立的 App ID 和 App Certificate。

根据以下步骤创建一个项目，同时获取 App ID 和 App Certificate。

1. 登录 <https://dashboard.agora.io>。
2. 在项目列表页面点击添加新项目。
3. 填写项目名。点击提交。



4. 启用 App Certificate。
 - a) 点击所创建项目右上角的编辑。
 - b) 点击 App Certificate 右方的启用按钮。仔细阅读关于 App Certificate 介绍后确认启用。

App Certificate: App Certificate尚未使用 **启用**

- c) 点击“眼睛”标志可查看 App Certificate，再次点击该标志可隐藏 App Certificate。

App Certificate:



注:

- 如您出于某种原因需更新 App Certificate，[请联系 support@agora.io](mailto:support@agora.io)。
- 请将 App Certificate 保存于服务器端，且对任何客户端均不可见。
- App Certificate 的用法，详见 [Channel Key 结构](#)。

集成 Channel Key 算法

开发者将生成 ChannelKey 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。

请登录以下网址获取示例代码：<https://github.com/AgoraLab/AgoraChannelKey>

如需要验证用户 ID(User ID), 详见下表所列的 ChannelKey 与 SDK 的版本兼容：

ChannelKey 版本	User ID(UID)	SDK 版本
ChannelKey4	指定用户的 UID	1.3 或更高版本
ChannelKey3	指定用户的 UID	1.2.3 或更高版本
ChannelKey	N/A	N/A

如不需验证 UID，则各版本 SDK 可使用任意 ChannelKey 版本，但建议升级至 ChannelKey4。

使用 Channel Key

每次客户端加入频道时：

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 App Certificate、App ID、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 Channel Key，返回给授权的客户端应用程序。
3. 客户端应用程序调用 AgoraRTC.Client.join 方法时，第一个参数要求为 Channel Key。

Agora 的服务器接收到 Channel Key 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 Channel Key 方案后，在用户进入频道时，用 Channel Key 或 Recording Key 替换原来的 App ID。

Channel Key 在一段时间后会失效，应用程序获知密钥失效（即当 onFailure 回调方法报告 ERR_CHANNEL_KEY_EXPIRED = 109 或 ERR_INVALID_CHANNEL_KEY = 110, 时），需调用 renewChannelKey() 方法进行更新。

Channel Key 安全性说明

采用 Channel Key 方案的企业，在用户加入频道时都需要提供此密钥，确保用户每次通话都

经过信令服务器的授权。Channel Key 采用 HMAC/SHA1 签名方案，提高了系统及通话的安全性。

Channel Key 结构

将下表中所有字段从前往后拼接：

字段	类型	长度	说明
版本号	字符串	3	Channel Key 算法的版本信息
签名	字符串	40	<p>签名的 hex 编码。将 App Certificate 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：</p> <ul style="list-style-type: none">✓ 服务类型：ASCII 可见字符串，由 Agora 提供。详见章节服务类型。✓ App ID：32 位 App ID 字符串。✓ 时间戳：Channel Key 和生成时的时间戳。✓ 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✓ 服务到期时间：用户在频道内截止通话的时间戳
App ID	字符串	32	App ID
授权时间戳	数字	10	Channel Key 生成时的时间戳，自 1970.1.1 开始到当前时间的秒数。授权该 Channel Key 在生成后的 5 分钟内可以加入频道，如果 5 分钟内没有加入频道，则该 Channel Key 无法再加入频道。
随机数	整数	8	32 位整数 hex 编码。随每个请求重新生成。

服务到期时间戳	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 AgoraRTC.client.join API 时，如果需要传入 Channel Key，需要使用这个类型来生成。 注： 设置服务到期时间并不意味着 Channel Key 失效，而仅仅用于限制用户在频道内的时间。例如某会议时间预计为 1 小时，会议结束后如果用户不退出频道，仍旧会被计费。设置到期时间，用户会在时间到期时自动被踢出频道。

签名算法：HMAC/SHA1

Channel Key 采用业界标准化的 HMAC/SHA1 加密方案，在 Node.js, Java, Python, C++ 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。

程序开发和部署

运行 Agora Native SDK for Web 的程序是标准的 JavaScript 程序，部署 JavaScript 只需加载 Agora JS 库并访问 SDK 里提供的 JS 延伸库。

加载 Agora JS 库 AgoraRtcAgentSDK-1.7.0.js, 可从 <https://rtc.sdk.agora.io/AgoraRtcAgentSDK-1.7.0.js> 或者 <http://rtc.sdk.agora.io/AgoraRtcAgentSDK-1.7.0.js> 地址下载。

运行示例网页程序

客户端

默认的示例程序只需要静态的 **App ID**（在示例程序网页上输入即可）。按照以下步骤运行示例程序：

1. 确保已安装本地网页服务器，如 Apache, NginX 或 Node.js。
2. 将 `./client/` 下的文件部署到网页服务器上。
3. 在网页服务器上用浏览器打开示例程序页面（关于推荐的浏览器，请参考章节 [浏览器支持](#)）。

如需使用 **Channel Key**，请执行以下操作：

注: 更多详细内容，请参考章节 [获取和使用 Channel Key](#)。

1. 设置和启动生成 Channel Key 的服务器。
2. 从客户端向生成 Channel Key 的服务器发送 http/https 请求，获取 Channel Key。
3. 在示例程序中调用 `AgoraRTC.client.join`。

服务器

如需使用更为安全的 **Channel Key**，可尝试以下代码。在实际使用时，开发者应将该方法应用到自己的服务器端程序上，并使用服务器上的现有的编程语言重新编程。本示例代码是 JavaScript 写成，因此需要标准的 Node.js 服务器上使用：

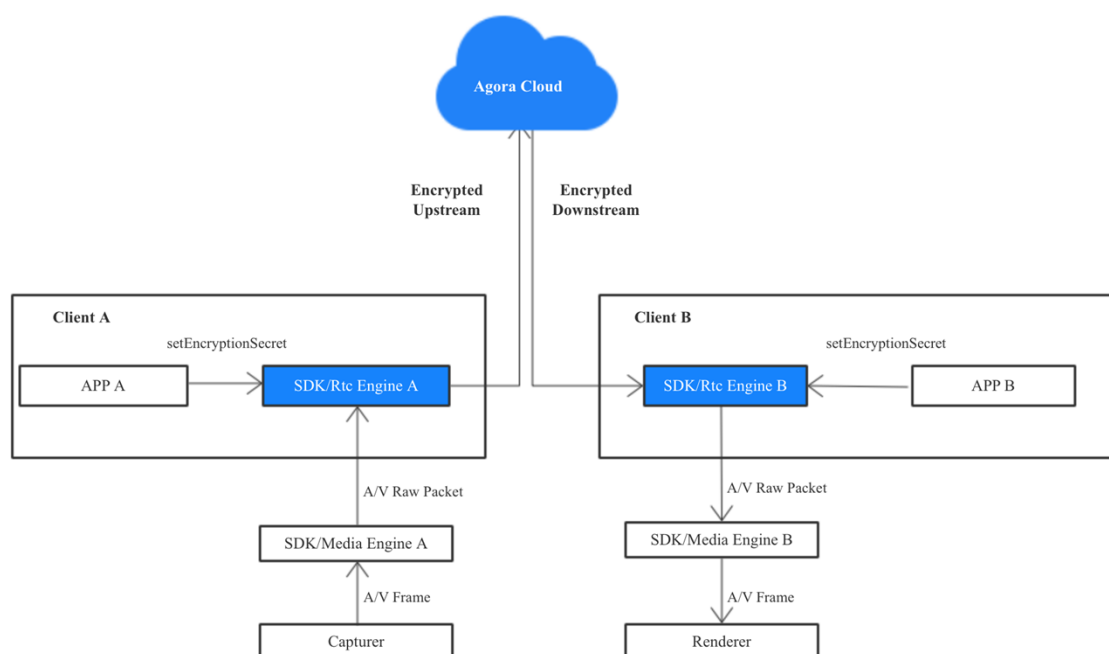
1. 在本地服务器或云主机上安装标准的 Node.js 服务器。
2. 在 `/server/nodejs/` 运行 'npm install'。
3. 在 `./server/nodejs/DemoServer.js` 下填写 APP_ID(旧称 VENDOR_KEY)和 APP_CERTIFICATE(旧称 SIGN_KEY)的值。用 'node DemoServer.js' 打开服务器。

加密数据

Agora SDK 允许您的应用程序使用 Agora SDK 的内置加密方案对语音和视频数据进行加密。

Agora SDK 使用 AES-128 和 AES-256 加密算法实现内置加密。您可调用 `setEncryptionSecret` 启用内置的加密功能，调用 `setEncryptionMode` 设置不同的加密模式。详见 [Agora Native SDK for Web – API 参考](#)。

下图描述了数据加解密流程：



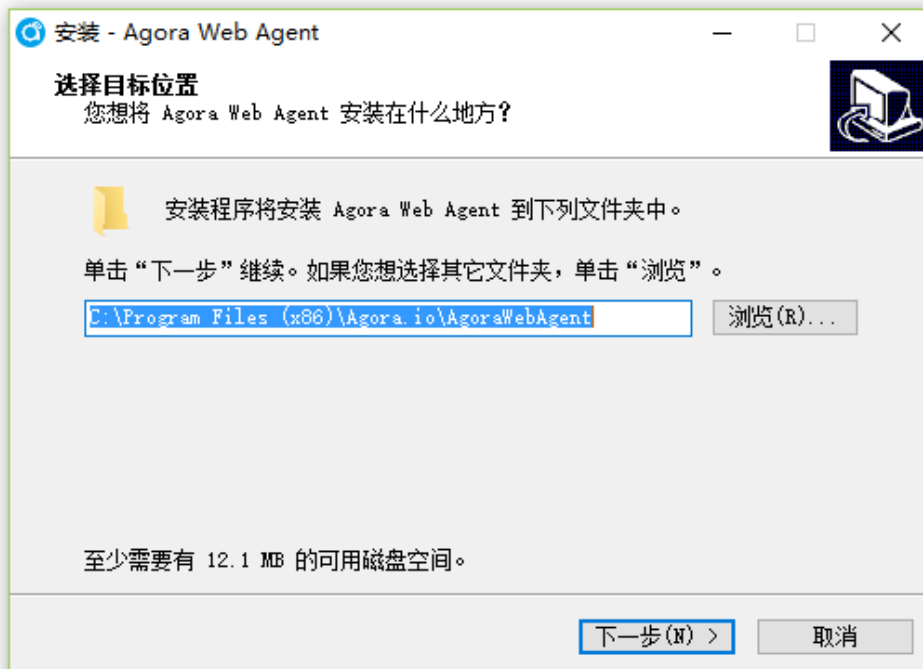
安装 AgoraWebAgent（Windows 或 Mac 平台）

完成部署之后，您（使用 Agora 服务的开发人员）的用户在使用客户端时，客户端将提示他们在 Windows 或者 Mac 上安装 AgoraWebAgent 应用程序。提示信息上显示的程序下载链接和安装指南均为 Agora 提供的。您可以直接给用户使用，或者将其定制化后让用户访问您的网页。

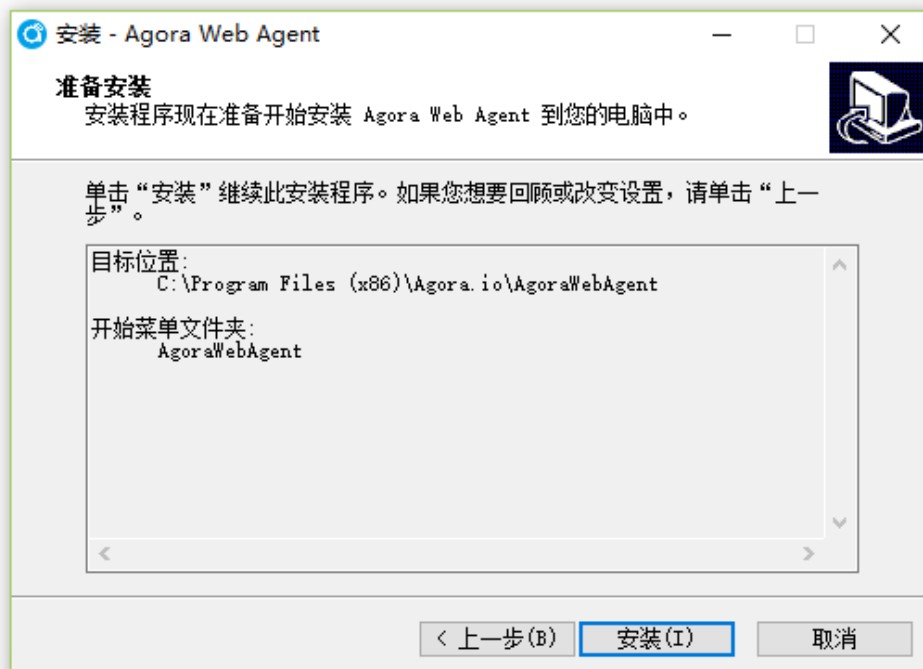
为了方便您定制化，以下再次提供了安装指南：

Windows 平台

1. 双击客户端提示信息里下载的 `AgoraWebAgent.exe` 安装程序。
2. 选择安装语言 **简体中文**，点击 **确认**。
3. 如果想更换安装目录，点击 **浏览**。否则，直接点击 **下一步**。



4. 点击安装，直到进度条显示安装完成。

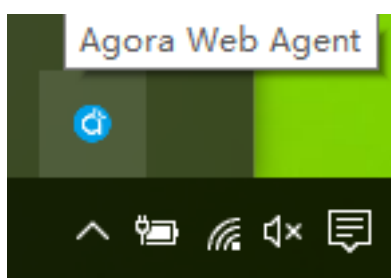


5. 选择 运行 **AgoraWebAgent.exe**，点击完成。



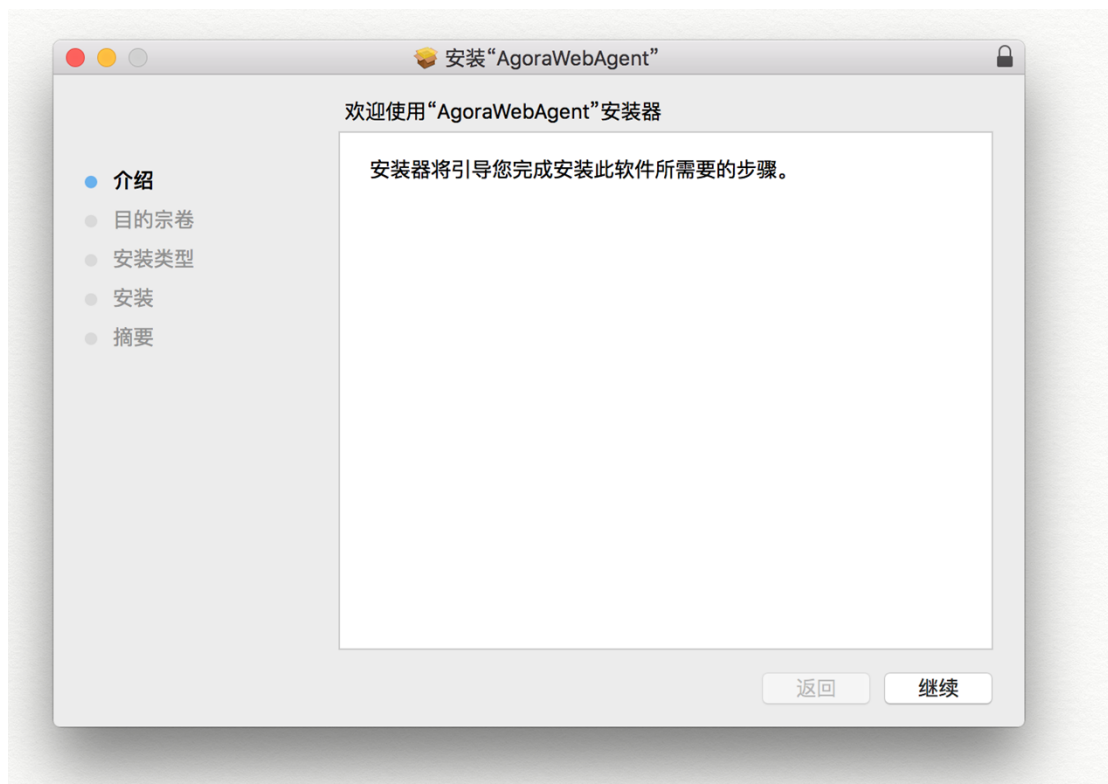
注:

AgoraWebAgent 安装完成后默认设置为开机自动启动，并在桌面上安装快捷方式，用户可以在任务管理器中设置允许或禁用开机自动启动。**AgoraWebAgent** 启动后图标会隐藏在桌面任务栏右下角。若使用过程中退出 **AgoraWebAgent**，可点击桌面快捷方式重新启动 **AgoraWebAgent**，重新启动后仍默认设置为开机自动启动。

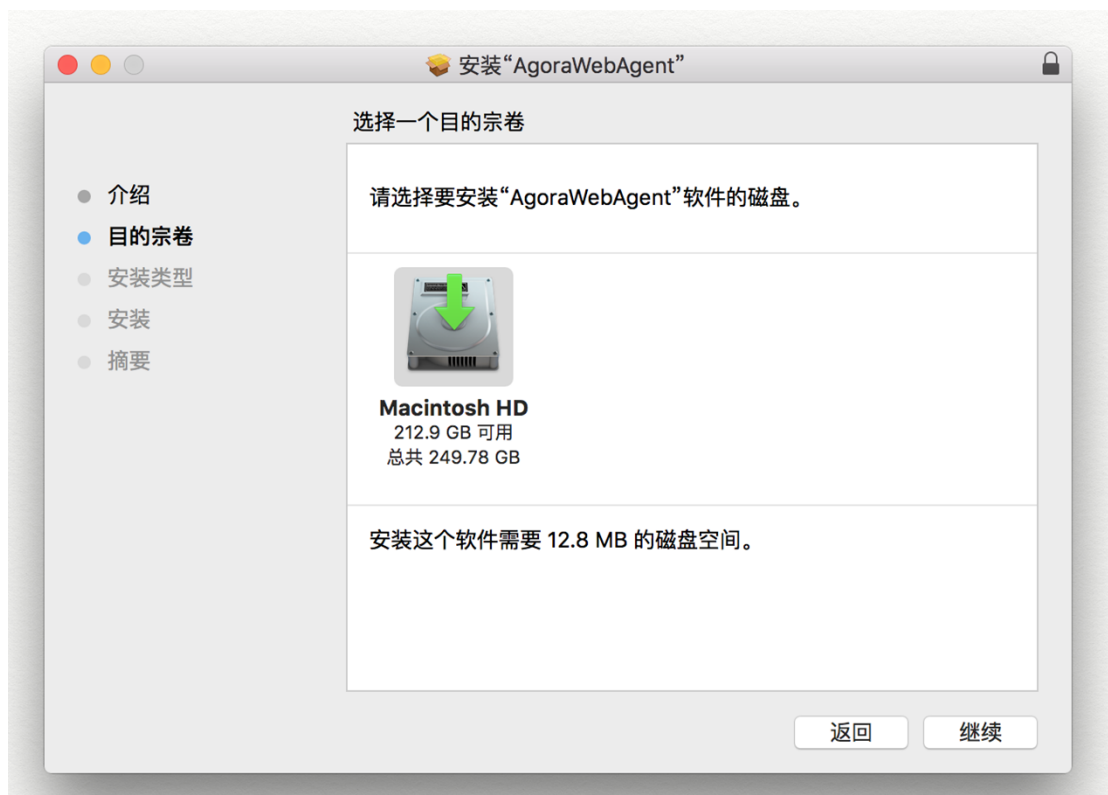


Mac 平台

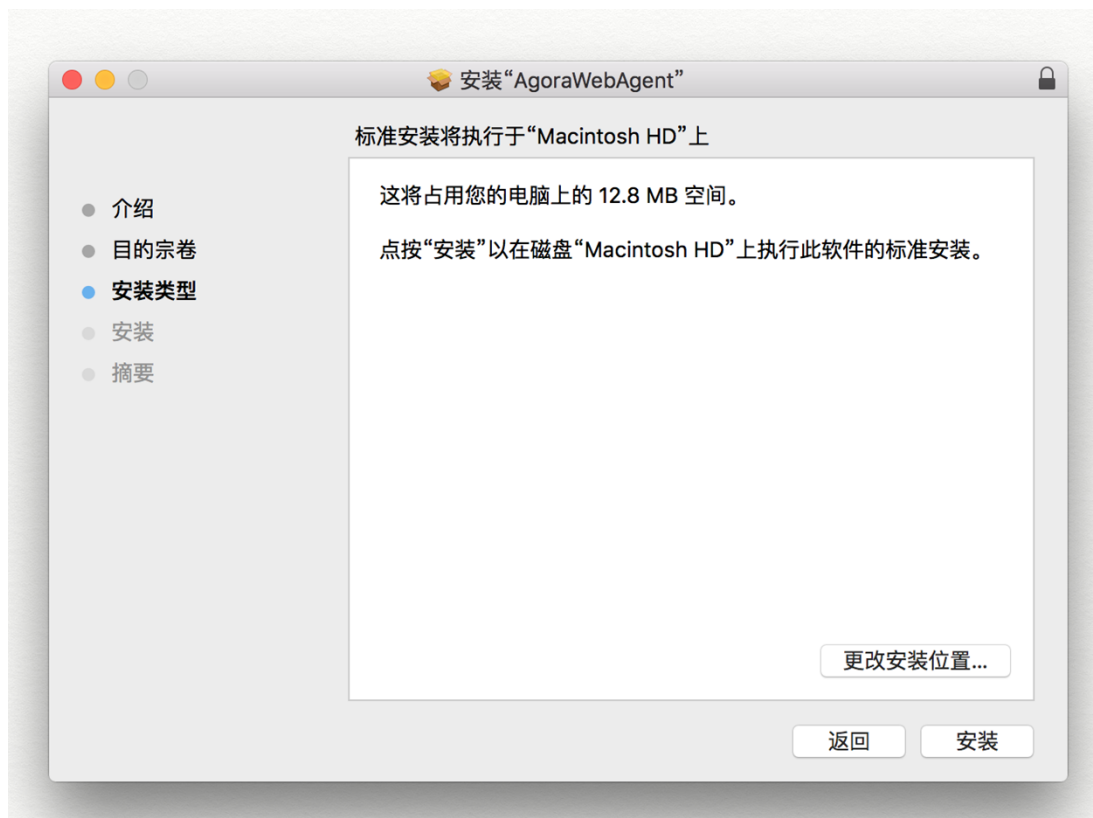
1. 双击客户端提示信息里下载的 **AgoraWebAgent.pkg** 安装程序。
2. 在安装界面点击**继续**。



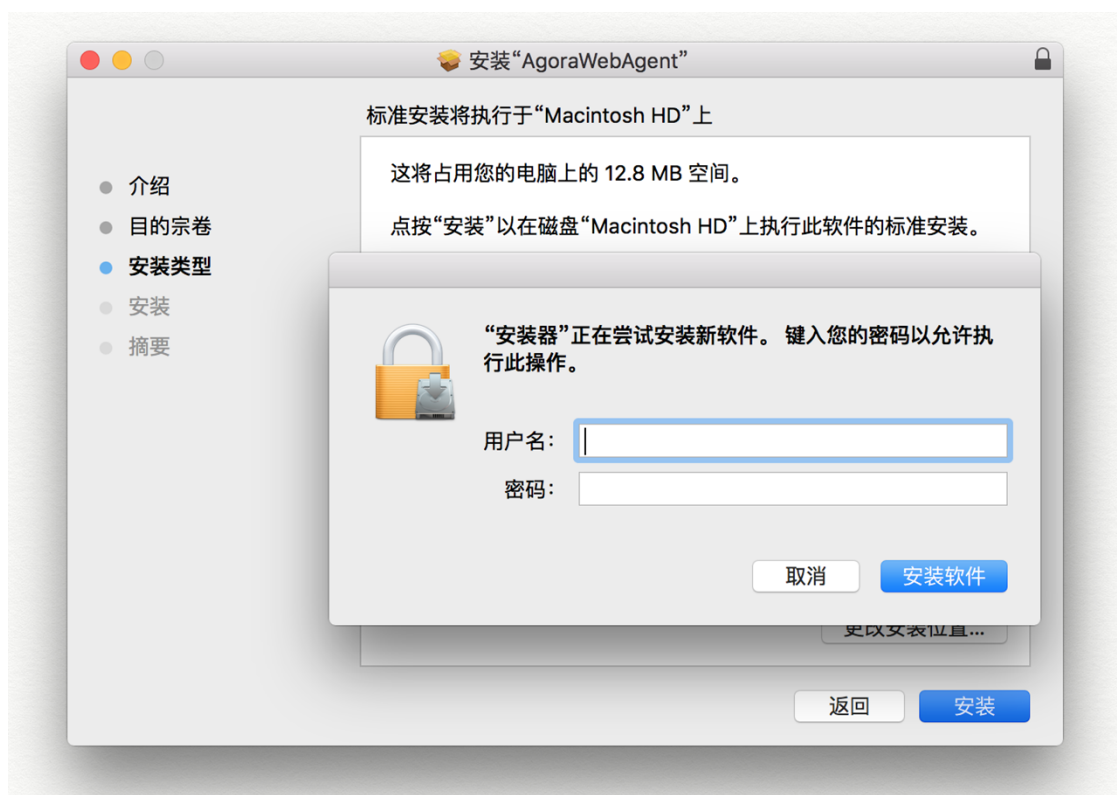
3. 点击继续。



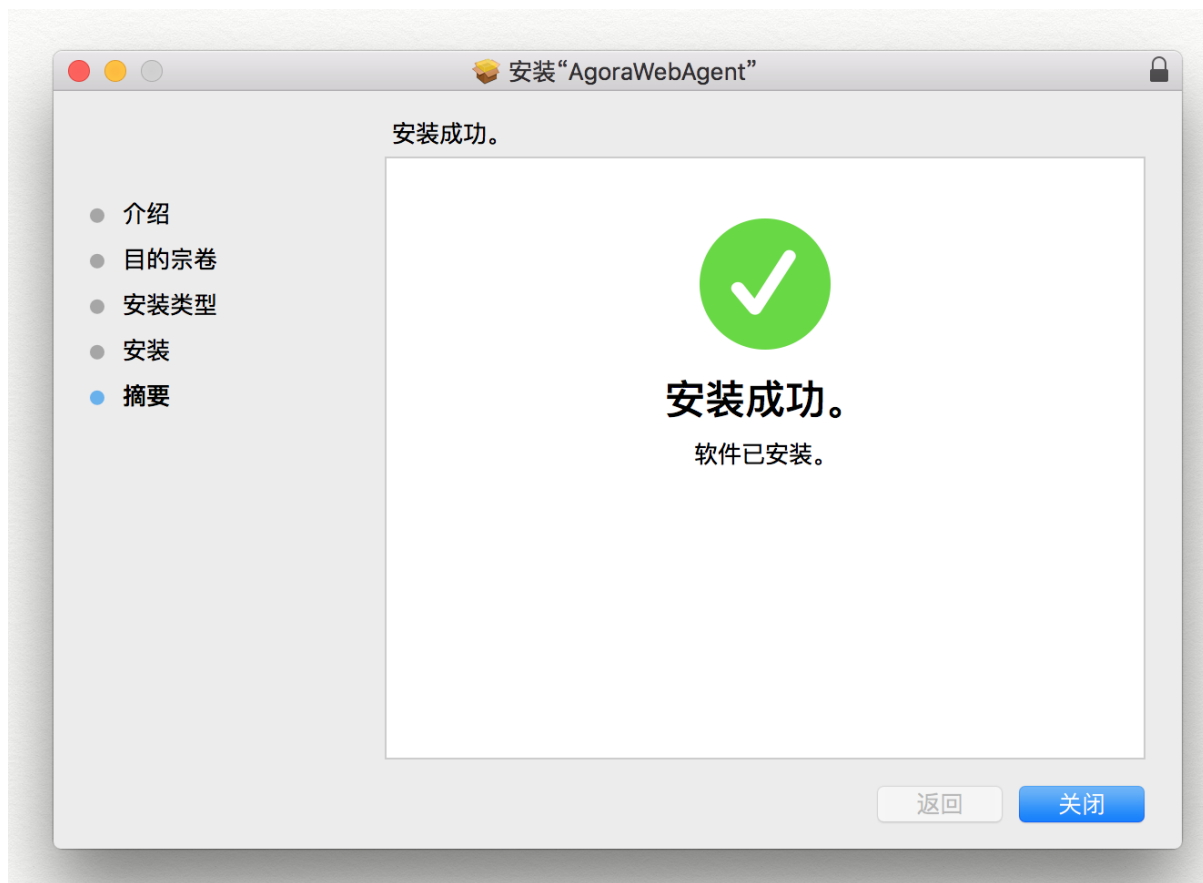
4. 点击安装在默认磁盘上安装程序。



5. 输入电脑的用户名和密码。点击**安装软件**。



6. 当出现以下消息时，安装完成。点击**关闭**。



注:

AgoraWebAgent 默认安装在应用程序中，默认设置为开机自动启动，用户可在上方工具栏中右键单击图标关闭或开启开机自动启动。若使用过程中退出 **AgoraWebAgent**，可单击应用程序中的 **AgoraWebAgent** 图标重新启动，重新启动后仍默认设置为开机自动启动。

卸载 AgoraWebAgent（Windows 或 Mac 平台）

Windows 平台

AgoraWebAgent 程序卸载流程跟在 Windows 上卸载一般程序相同，都在控制面板里卸载。以下选取 Windows 8 为例：

1. 进入桌面>打开控制面板。
2. 在程序下面点击卸载程序。
3. 找到 AgoraWebAgent.exe 右键选择卸载直到卸载完成。

Mac 平台

1. 在 **Docker** 上点击 **Finder**。
2. 打开应用程序文件夹。

3. 找到 AgoraWebAgent 右键选择移到废纸篓。

Agora Native SDK for Web - API 参考

Agora Web SDK 库包含以下三种接口类：

AgoraRTC	使用 AgoraRTC 对象创建客户端和音视频流对象。
Client	提供 AgoraRTC 核心功能的 web 客户端对象。
Stream	通话中的本地或远程音视频流。

AgoraRTC 接口类

创建客户端对象(createRtcClient)

createRtcClient()

创建并返回客户端对象。单次通话中仅需调用一次。

示例代码：

```
var client = AgoraRTC.createRtcClient();
```

创建本地音视频流对象(createStream)

createStream(spec)

创建并返回本地音视频流对象。

参数名称	类型	描述
spec	对象	<p>该对象包含以下属性：</p> <ul style="list-style-type: none">● streamID: 音视频流 ID，通常设置为 uid，可通过 Client.join 回调方法获取。● local: 是否本地音视频流，必须指定为 true。远端音视频流在 SDK 内部创建，不需要指定 false。● video: 是否启用视频(打开摄像头)。true 表示启用视频，会打开摄像头。false 表示禁用视频，不开摄像头。

示例代码：

```
var stream = AgoraRTC.createStream({ streamID: uid, local: true, video: true});
```

Client 接口类

代表提供核心 AgoraRTC 功能的网页客户端对象。

方法:

初始化客户端对象(init)

init(appld, onSuccess, onFailure)

初始化客户端对象。

参数名称	类型	描述
appld	字符串	注册时由Agora.io提供的App ID。对于大多数对安全要求不高的开发环境来说，用App ID 访问Agora Global Network提供的服务即可。
onSuccess	函数	(非必选项) 当方法成功时被调用的函数。
onFailure	函数	<p>(非必选项) 客户端对象生命期中出现错误时执行的回调函数。</p> <p>当 onFailure 回调中出现 CLOSE_BEFORE_OPEN 时，表明无法连接到 AgoraWebAgent。可能的原因有：</p> <ul style="list-style-type: none">• 无可可用网络。由于采用了安全的 websocket 本地连接，连接 AgoraWebAgent 需要 DNS 服务，当 DNS 服务不可用时连不上 AgoraWebAgent。• 未启动 AgoraWebAgent。• 未安装 AgoraWebAgent。• 使用了代理，导致无法连接 AgoraWebAgent。 <p>当 onFailure 回调中出现 INCOMPATIBLE_WEBAGENT 时，可能的原因是 1.6 版的 JS 与之前版本的 AgoraWebAgent 不兼容，需根据回调里返回的 URL 升级 AgoraWebAgent。</p>

示例代码:

```
client.init(appld, function() {  
    log("client initialized");  
    //join channel  
    .....  
})
```

```

    }, function(err) {
        log("client init failed ", err);

        //error handling

    });

```

加入 AgoraRTC 频道(join)

join(channelKey, channelName, uid, onSuccess, onFailure)

该方法让用户加入 AgoraRTC 频道。

参数名称	类型	描述
channelKey	字符串	<p>对于安全有极高要求的使用者需要使用Channel Key，否则可使用App ID。Channel Key和App ID的切换可以在Dashboard操作。</p> <p>Channel Key 是通过 Agora 提供的安全算法生成，所需的 App ID 和 App Certificate 可以在 Dashboard 获取。</p>
channelName	字符串	<p>标识通话的频道名称，长度在 64 字节以内的字符串。</p> <p>以下为支持的字符集范围（共 89 个字符）：</p> <p>a-z</p> <p>A-Z</p> <p>0-9</p> <p>空格</p> <p>!#\$%&</p> <p>()+, -</p> <p>::<=.</p> <p>>? @[]</p> <p>^ _ `</p> <p>{ }</p> <p>~</p>
uid	整数	<p>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），服务器会自动分</p>

参数名称	类型	描述
		配一个，并在 <code>onSuccess</code> 回调方法中返回。
<code>onSuccess</code>	函数	（非必选项）方法调用成功时执行的回调函数，返回值为代表用户身份的 <code>uid</code> 。
<code>onFailure</code>	函数	（非必选项）方法调用失败时执行的回调函数。

示例代码:

```
client.join(key, '1024', 0, function(uid) {
    log("client " + uid + " joined channel");
    //create local stream
    .....
}, function(err) {
    log("client join failed ", err);
    //error handling
});
```

更新 Channel Key(`renewChannelKey`)

`renewChannelKey(channelKey,onSuccess,onFailure)`

用户更新 Channel Key。参数解释同上。

当调用 `client.init` 时传入的 `onFailure` 回调中出现以下信息时，

`ERR_CHANNEL_KEY_EXPIRED = 109,`

`ERR_INVALID_CHANNEL_KEY = 110,`

用户需要在 APP 上调用 `renewChannelKey` 来更新 Channel Key。

参数名称	类型	描述
<code>channelKey</code>	string	Channel Key. 详见 获取和使用 Channel Key 。
<code>onSuccess</code>	函数	（非必选项）方法调用成功时执行的回调函数。
<code>onFailure</code>	函数	（非必选项）方法调用失败时执行的回调函数。

离开 AgoraRTC 频道(leave)

leave(onSuccess, onFailure)

该方法让用户离开 AgoraRTC 频道。

参数名称	类型	描述
onSuccess	函数	(非必选项) 方法调用成功时执行的回调函数。
onFailure	函数	(非必选项) 方法调用失败时执行的回调函数。

示例代码:

```
client.leave(function() {  
    log("client leaves channel");  
    .....  
}, function(err) {  
    log("client leave failed ", err);  
    //error handling  
});
```

上传音视频流(publish)

publish(stream, onSuccess, onFailure)

将本地音视频流上传至服务器。

参数名称	类型	描述
stream	对象	本地音视频流对象。
onSuccess	函数	(非必选项) 方法调用成功时执行的回调函数。
onFailure	函数	(非必选项) 方法调用失败时执行的回调函数。

示例代码 :

```
client.publish(stream, function(err) {  
    console.log ("stream published");  
});
```

```

.....
}, function (err) {
    console.log ("failed to publish stream");
});

```

取消上传音视频流 (unpublish)

unpublish(stream, onSuccess, onFailure)

取消上传本地音视频流

参数名称	类型	描述
stream	对象	本地音视频流对象。
onSuccess	函数	（非必选项）方法调用成功时执行的回调函数。
onFailure	函数	（非必选项）方法调用失败时执行的回调函数。

示例代码:

```

client.unpublish(stream, function(err) {
    console.log("stream unpublished");
}, function (err) {
    console.log("failed to unpublish stream");
});

```

接收远程音视频流(subscribe)

subscribe(stream, onFailure)

从服务器端接收远程音视频流。

参数名称	类型	描述
stream	对象	远程音视频流对象。
onFailure	函数	（非必选项）方法调用失败时执行的回调函数。

示例代码:

```

client.subscribe(stream, function(err) {
    log("stream unpublished");
    .....
})

```

取消接收远程音视频流 (unsubscribe)

unsubscribe(stream, onFailure)

该方法取消接收远程音视频流。

参数名称	类型	描述
stream	对象	远程音视频流对象。
onFailure	函数	（非必选项）方法调用失败时执行的回调函数。

示例代码:

```

client.unsubscribe(stream, function(err) {
    log("stream unpublished");
    .....
})

```

枚举平台设备 (getDevices)

getDevices(callback)

该方法枚举平台摄像头耳麦设备。注：Mac 上暂不支持。

参数名称	类型	描述
callback	函数	<p>用回调函数来获取设备信息</p> <p>例如 callback(devices):</p> <p>devices[0].deviceId: device ID</p> <p>devices[0].label: device name in string</p> <p>devices[0].kind: 'audioinput', 'videoinput'</p>

示例代码:

```
client.getDevices (function(devices) {  
    var dev_count = devices.length;  
    var id = devices[0].deviceId;  
});
```

启用内置的加密密码(setEncryptionSecret)

function setEncryptionSecret(secret)

在加入频道之前，应用程序需调用 `setEncryptionSecret` 指定 `secret` 来启用内置的加密功能，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	加密密码
Return Value	0: 方法调用成功 <0: 方法调用失败

示例代码:

```
client.setEncryptionSecret("1234");
```

设置内置的加密方案(setEncryptionMode)

function setEncryptionMode (encryptionMode)

Agora SDK 支持内置加密功能，默认使用 AES-128-XTS 加密方式。如需使用其他加密方式，可以调用该 API 设置。同一频道内的所有用户必须设置相同的加密方式和 `secret`，才能进行通话。关于这几种加密方式的区别，请参考 AES 加密算法的相关资料。

注：在调用本方法前，需要先调用 `setEncryptionSecret` 启用内置加密功能。

名称	描述
encryptionMode	加密方案。目前支持以下几种： <ul style="list-style-type: none"> “aes-128-xts”：128 位 AES 加密，XTS 模式 “aes-256-xts”：256 位 AES 加密，XTS 模式 “”：设置为空字符串时，使用默认加密方式“aes-128-xts”。
Return Value	0： 方法调用成功 <0： 方法调用失败

示例代码:

```
client.setEncryptionMode ("aes-256-xts");
```

选择设备 (selectDevice)

selectDevice(device)

在当前对话中选择使用语音/视频设备。

参数名称	类型	描述
device	对象	从 getDevices 返回的设备清单里选择设备。Agora 会在当前对话中使用已选择语音/视频设备。

示例代码:

```
client.selectDevice(device);
```

开始录制 (startRecording)

startRecording(recordingKey, onSuccess, onFailure)

请求 ARS 开始录制会话中的所有音视频。

参数名称	类型	描述
recordingKey	字符串	用于录制的密钥（Recording Key）。后端应用程序必须集成录制的 Recording Key 算法授权用户访问 Agora Recording Server。
onSuccess	函数	方法调用成功时执行的回调函数。

参数名称	类型	描述
onFailure	函数	方法调用失败时执行的回调函数。

示例代码:

```
client.startRecording(key, function(data) {
    console.log("start recording successfully.");
}, function(err) {
    console.log("failed to start recording" + err);
});
```

停止录制(stopRecording)

stopRecording(recordingKey, onSuccess, onFailure)

请求 ARS 停止录制会话中的音视频。stopRecording 和 startRecording 使用相同的 Recording Key, 且 SDK 会储存该密钥。

参数名称	类型	描述
recordingKey	字符串	用于录制的密钥 (Recording Key)。后端应用程序必须集成录制的 Recording Key 算法授权用户访问 Agora Recording Server。
onSuccess	函数	方法调用成功时执行的回调函数。
onFailure	函数	方法调用失败时执行的回调函数。

示例代码:

```
client.stopRecording(key, function(data) {
    console.log("stop recording successfully.");
}, function(err) {
    console.log("failed to stop recording" + err);
});
```

查询录制状态(queryRecordingStatus)

queryRecordingStatus (onStatus)

向 ARS 查询音视频录制状态。

参数名称	类型	描述
onStatus	函数	返回录制状态的回调函数。

示例代码:

```
client.queryRecordingStatus(function(result) {  
    console.log(result);  
    switch(result.status) {  
        case 0:  
            // TODO:...  
            break;  
        default:  
            break;  
    }  
});
```

获取桌面窗口列表(getWindows)

getWindows (callback)

该方法用于获取操作系统的桌面窗口。应用程序可以用此方法获取窗口列表，然后选择指定要共享的窗口。

名称	类型	描述
callback	函数	用回调函数来获取窗口列表。例如: callback(windows): windows[0].windowId: window id in string windows[0].title: window title in string

开启屏幕共享(startScreenSharing)

startScreenSharing (window, onSuccess, onFailure)

该方法让用户实现屏幕共享。共享的屏幕或窗口由窗口句柄 **window** 指定。启用屏幕共享功能后，对方屏幕上将显示您共享的桌面或窗口，而不是您摄像头拍摄的视频。

名称	类型	描述
window	字符串	窗口 ID。当 windowId="" 时，共享整个屏幕。
onSuccess	函数	方法调用成功时执行的回调函数。
onFailure	函数	方法调用失败时执行的回调函数。

指定共享窗口(setScreenSharingWindow)

setScreenSharingWindow (window, onSuccess, onFailure)

该方法让用户在启用屏幕共享功能后切换共享屏幕或特定窗口。共享的屏幕或窗口由窗口句柄 window 指定。

名称	类型	描述
window	字符串	窗口 ID。当 windowId="" 时，共享整个屏幕。
onSuccess	函数	方法调用成功时执行的回调函数。
onFailure	函数	方法调用失败时执行的回调函数。

停止屏幕共享(stopScreenSharing)

int stopScreenSharing();

该方法关闭屏幕共享功能。

名称	类型	描述
onSuccess	函数	方法调用成功时执行的回调函数。
onFailure	函数	方法调用失败时执行的回调函数。

清理资源(close)

close();

该方法清理 client 里的所有资源。一经调用，用户无法再调用任何其他方法，包括 init。如需再次使用 client 对象，需重建该对象，并调用 init 进行初始化。

示例代码：

```
var client = AgoraRTC.createRtcClient();
client.init(appID, onInitClientSuccess, onInitClientFailure);
client.close();
client = undefined;

// create a new client
client = AgoraRTC.createRtcClient();
client.init(appID, onInitClientSuccess, onInitClientFailure);
```

事件:

远程音视频流已添加事件(**stream-added**)

通知应用程序远程语音/视频流已添加。

示例代码:

```
client.on('stream-added', function(evt) {  
    var stream = evt.stream;  
    log("new stream added ", stream.getId());  
    //subscribe the stream  
    .....  
})
```

远程音视频流已订阅事件(**stream-subscribed**)

通知应用程序已订阅远程语音/视频流，可以开始播放远端视频。

示例代码:

```
client.on('stream-subscribed', function(evt) {  
    var stream = evt.stream;  
    log("new stream subscribed ", stream.getId());  
    //play the stream  
    .....  
})
```

远端用户已离开房间事件(**peer-leave**)

通知应用程序远端用户已离开房间 (例如对方用户调用了 `client.leave()`)，可以停止播放对方的视频。

示例代码:

```
client.on('peer-leave', function(evt) {  
    var uid = evt.uid;  
    log("remote user left ", uid);  
})
```

```
.....  
    })
```

远端用户视频发送事件(peer-mute-video)

通知应用程序远端用户已选择不发送或者重新发送视频流。

参数名称	描述
muted	当 muted 为 true 时，表示远端用户选择不发送视频，即其他用户无法看见其图像。 当 muted 为 false 时，表示远端用户选择发送视频，即其他用户可以看见其图像。

示例代码：

```
client.on("peer-mute-video", function (event) {  
    var message = event.msg;  
    if (message.muted) {  
        console.log("remote user " + message.uid + " muted video");  
    } else {  
        console.log("remote user " + message.uid + " unmuted video");  
    }  
}
```

远端用户设置静音事件(peer-mute-audio)

通知应用程序远端用户已选择不发送或重新发送语音流。

参数名称	描述
muted	当 muted 为 true 时，表示远端用户已设置静音，即其他用户无法听见其声音。 当 muted 为 false 时，表示远端用户选择重新发送语音流，即其他用户可以听见其声音。

示例代码:

```
client.on("peer-mute-audio", function (event) {
    var message = event.msg;
    if (message.muted) {
        console.log("remote user " + message.uid + " muted audio");
    } else {
        console.log("remote user " + message.uid + " unmuted audio");
    }
}
```

网络连接中断或丢失事件(error)

在 SDK 和服务端失去了网络连接后，会触发 **CONNECTION_INTERRUPTED** 回调事件，并尝试自动重连。在一定时间内（默认 10 秒）如果没有重连成功，触发 **CONNECTION_LOST** 回调。除非 APP 主动调用 **leave**，SDK 仍然会自动重连。

更多错误代码说明，详见[附录](#)。

示例代码:

```
client.on("error", function (event) {
    var message = event.msg;
    if (message.reason === "CONNECTION_INTERRUPTED") {
        console.log("connection interrupted");
    } else if (message.reason === "CONNECTION_LOST") {
        console.log("connection lost");
    }
}
```

Stream 接口类

初始化本地音视频对象(init)

init(onSuccess, onFailure)

此方法初始化本地音视频流对象，远端音视频不需要应用程序初始化，SDK 内部会处理。

参数名称	类型	描述
onSuccess	函数	（非必选项）方法调用成功时执行的回调函数。
onFailure	函数	（非必选项）方法调用失败时执行的回调函数。

示例代码:

```
stream.init(function() {
    log("local stream initialized");
    // publish the stream
    .....
}, function(err) {
    log("local stream init failed ", err);
    //error handling
});
```

关闭视频流(close)

close()

此方法关闭视频流，用户在网页上不会再看到这一路视频。

播放音/视频流(play)

play(elementID, onFailure)

播放视频流或音频流

参数名称	类型	描述
elementID	字符串	html 元素 ID。
onFailure	函数	(非必选项)

示例代码:

```
stream.play(tdiv_id).play(onFailure){
    if (err) {
        console.log('failed to play stream');
    }
}
```

```
});
```

停止播放视频流(stop)

stop()

停止在网页上播放视频流。

启用音频流(enableAudio)

enableAudio(callback)

启用音频流，完成后会调用 callback。

禁用音频流(disableAudio)

disableAudio(callback)

禁用音频流，完成后会调用 callback。

启用视频流(enableVideo)

enableVideo(callback)

启用视频流，完成后会调用 callback。如果继续收到该用户的视频流，应用程序会收到 stream-added 事件，应用程序可以重新播放视频。

禁用视频流(disableVideo)

disableVideo(callback)

禁用视频流，完成后会调用 callback。

获取用户 ID(getId)

getId()

获取音视频流 ID，即对应的用户 ID。

设置视频编码属性(setVideoProfile)

setVideoProfile(profile)

设置视频编码属性，为非必选项。每个属性对应一套视频参数，如分辨率、帧率、码率等。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。

参数 profile 是一个 json 对象，包含两个属性：

名称	描述
profile	视频属性。详见下面的视频属性定义。

swapWidthAndHeight	<p>是否交换宽和高。</p> <p>true: 交换宽和高</p> <p>false: 不交换宽和高(默认)</p>
--------------------	---

注:

- 应在调用 `AgoraRTC.client.join` 进入频道前设置视频属性。
- 该方法仅设置编码器编出的码流属性，可能跟最终显示的属性不一致，例如编码码流分辨率为 **640x480**，码流的旋转属性为 **90 度**，则显示出来的分辨率为竖屏模式。

视频属性 (Profile) 定义

视频 Profile (字符串)	枚举值	分辨率(宽 x 高)	帧率(fps)
"120P"	0	160x120	15
"120P_2"	1	120x160	15
"120P_3"	2	120x120	15
"180P"	10	320x180	15
"180P_2"	11	180x320	15
"180P_3"	12	180x180	15
"240P"	20	320x240	15
"240P_2"	21	240x320	15
"240P_3"	22	240x240	15
"360P"	30	640x360	15
"360P_2"	31	360x640	15
"360P_3"	32	360x360	15
"360P_4"	33	640x360	30
"360P_5"	34	360x640	30
"360P_6"	35	360x360	30
"480P"	40	640x480	15
"480P_2"	41	480x640	15
"480P_3"	42	480x480	15
"480P_4"	43	640x480	30
"480P_5"	44	480x640	30
"480P_6"	45	480x480	30
"480P_7"	46	640x480	15
"720P"	50	1280x720	15
"720P_2"	51	720x1280	15
"720P_3"	52	1280x720	30
"720P_4"	53	720x1280	30

示例代码：

```
stream.setVideoProfile('480P');
```

附录

Agora SDK 在调用 API 或运行时，可能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_APP_ID	101	指定的 App ID 无效。
ERR_INVALID_CHANNEL_NAME	102	指定的频道名无效。
ERR_CHANNEL_KEY_EXPIRED	109	当前使用的 Channelkey 过期，不再有效。
ERR_INVALID_CHANNEL_KEY	110	指定的 Channel key 无效。一般是因为 key 生成的不对。
ERR_CONNECTION_INTERRUPTED	111	网络连接中断。 CONNECTION_INTERRUPTED 的回调。仅用于 Agora Native SDK for Web。
ERR_CONNECTION_LOST = 112	112	网络连接丢失。CONNECTION_LOST 的回调。仅用于 Agora Native SDK for Web。
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_ADM_HOWLING	1051	音频处理模块:

		检测到啸叫。
--	--	--------

Agora CaaS, Agora Global Network, Agora Native SDK和Agora Web SDK为Agora.io的注册商标。Agora.io经营业务中也使用Agora Lab这一名称。本文档中提及的其他产品或公司名称均为其各自公司的注册商标。

©2016 Agora.io. 版权所有。