



Agora Signaling SDK for Mac

参考手册

1.0 版

support@agora.io

目录

简介	5
Agora 实时云	5
Agora Signaling SDK for Mac	5
用户须知	6
开发环境要求	6
所需 SDK	6
所需文档	6
功能介绍	6
账号消息系统	6
概览	6
消息队列	7
用户登录	7
用户属性	7
事件与查询	7
纯文本消息	7
频道系统	7
概览	7
加入频道	7
频道事件	8
频道消息	8
频道属性	8
呼叫系统	8
概览	8
呼叫流程	8
呼叫超时	9
接入方式	9
SDK 接入	10
Server API 接入	10
SIP 网关接入	10
获取和使用 Signaling Key	11
获取 App ID 和 App Certificate	11
集成 Signaling Key 算法	12
使用 Signaling Key	12
Signaling Key 的安全性	12
Signaling Key 结构	12
集成和初始化 SDK	13
Signaling SDK API 参考	15
方法	15
登录(login)	15

退出(login).....	15
加入频道(channelJoin)	15
离开频道(channelLeave).....	16
查询频道用户数(channelQueryUserNum).....	16
设置频道属性(channelSetAttr)	16
删除频道属性(channelDelAttr)	16
删除所有频道属性(channelClearAttr)	17
发起呼叫(channelInviteUser)	17
发起呼叫(channelInviteUser2)	17
向远端发送 DTMF 消息(channelInviteDTMF).....	18
接受呼叫(channelInviteAccept)	18
拒绝呼叫(channelInviteRefuse).....	18
关闭呼叫(channelInviteEnd).....	19
发送点对点消息(messageInstantSend).....	19
发送频道消息(messageChannelSend).....	19
设置用户属性(setAttr).....	20
获取自己的用户属性(getAttr).....	20
获取自己的所有用户属性(getAttrAll)	20
获取某个用户的用户属性(getUserAttr).....	20
获取某个用户的所有用户属性(getUserAttrAll).....	20
检查当前是否在线(isOnline).....	21
内测接口和过期接口	21
回调事件.....	22
连接丢失回调(onReconnecting)	22
重连成功回调(onReconnected)	22
登录成功回调(onLoginSuccess).....	22
退出登录回调(onLogout)	22
登录失败回调(onLoginFailed)	22
加入频道回调(onChannelJoined)	23
加入频道失败回调(onChannelJoinFailed)	23
离开频道回调(onChannelLeaved).....	23
其他用户加入频道回调(onChannelUserJoined)	23
其他用户离开频道回调(onChannelUserLeaved).....	24
获取频道内用户列表回调(onChannelUserList)	24
返回查询的用户数量回调(onChannelQueryUserNumResult)	24
频道属性发生变化回调(onChannelAttrUpdated)	24
收到呼叫邀请回调(onInviteReceived)	25
远端已收到呼叫回调(onInviteReceivedByPeer).....	25
远端已接受呼叫回调(onInviteAcceptedByPeer).....	25
对方已拒绝呼叫回调(onInviteRefusedByPeer)	26
呼叫失败回调(onInviteFailed).....	26
对方已结束呼叫回调(onInviteEndByPeer)	26
本地已结束呼叫回调(onInviteEndByMyself).....	26
消息发送失败回调(onMessageSendError).....	27
消息已发送成功回调(onMessageSendSuccess).....	27

收到用户消息回调(onMessageInstantReceive)	27
收到频道消息回调(onMessageChannelReceive)	28
已打印日志回调(onLog)	28
已获取用户属性查询结果回调(onUserAttrResult)	28
已获取所有用户属性查询结果回调(onUserAttrAllResult)	28
内测接口和过期接口	29
错误代码	30

简介

Agora 实时云

Agora 实时云运用 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:

SDK	Android	iOS	Windows	Mac	Web
Agora Native SDK	√	√	√	√	√
Agora Signaling SDK	√	√	√	√	

Agora Native SDK 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。Agora Signaling SDK 提供可靠的音视频控制服务。

本文主要介绍如何使用 Agora Signaling SDK for Mac。

Agora Signaling SDK for Mac

您可以编程实现以下功能(详见[功能介绍](#)):

- **账号消息系统**: 账号消息系统为频道系统和呼叫系统的基础，用户登录之后，能可靠地收发消息，开展其他业务。
- **频道系统**: 支持众多用户同时在线，并提供可靠的频道事件、属性和消息。典型的应用场景包括：音视频会议的消息、控麦、其他控制以及视频直播的频道消息等。
- **呼叫系统**: 提供音视频呼叫和对外的 SIP 呼叫，包括混合 VOIP 用户和 PSTN 电话用户的混合呼叫会议。

Agora Signaling SDK 提供 API 实现本文所述的所有功能，详见 [Signaling SDK API 参考](#)。

Agora Signaling SDK 在调用 API 或运行时可能会返回错误或警告消息。详见[错误代码](#)。

用户须知

开发环境要求

根据[接入方式](#)所述，您可以通过三种方式接入 Agora 信令系统，其中通过 Agora SDK 接入的系统和网络要求如下：

平台	系统要求	网络要求
OS X	v10.6 "Snow Leopard"或以上	HTTP 80 端口, TCP 8181 端口

所需 SDK

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

所需文档

- **本参考手册**：必读
- 如想使用 **Agora SDK** 实现应用程序的音视频通信功能，请结合所在平台的 **Agora Native SDK** 参考手册一起使用。所有 **SDK** 以及参考手册均可通过 <http://cn.agora.io/download/> 下载。

功能介绍

账号消息系统

本章介绍账户消息系统的详细功能和注意事项：

概览

Agora 信令系统账户用于识别用户身份，且用户的账号无需注册：

- 使用 **SDK** 接入：用户用厂商生成的 **Signaling Key** 登录即可。详见[获取和使用 Signaling Key](#)。
- 使用 **Server API** 接入：免登录。
- 使用 **SIP** 网关接入：联系 support@agora.io。

详见[接入方式](#)。

消息队列

所有用户均有自己的消息队列：

- 不管用户是否在线，消息都不会丢失。
- 消息可能会超时，且不同消息的超时时限不尽相同。

用户登录

用户只有在登录信令系统之后，才能进行各项操作和收发消息：

登录之后，可能会有暂时的网络中断，这期间无法收取消息，也无法进行任何操作。当网络恢复后，用户能收取消息，且消息不会丢失，除非超时。如果网络中断时间过长，**SDK** 则会触

发 Logout 事件。此时服务端也会将用户标记为 Logout 状态。用户须重新调用 Login 才可以继续操作。

目前一个账号只能在一个设备上登录，不能同时在多个设备同时登录。每次登录，均会从上一次登录的设备退出（SDK 触发 onLogout 事件）。

登录时须使用 Signaling Key，详见[获取和使用 Signaling Key](#)。

用户属性

所有用户均有自己的属性表：

用户可以设置一些属性，自己或者其他用户都可以读取到这些属性。属性可以用于存储用户的头像地址和昵称等信息。

事件与查询

- 用户均可查询其他用户的在线状态、属性。
- 用户均可订阅其他用户的在线状态、属性变化(只支持 Server API)，详见 **Server API 参考手册**。

纯文本消息

用户之间可以发送纯文本的简单消息。例如文字、网址、图像地址、json 等内容。

频道系统

概览

信令的频道，基于可靠的账号消息系统，确保不会丢失消息。用户可以基于这个前提，来构造其上的业务系统。

当网络暂时中断时，无法收到消息，网络恢复时，期间的消息又会重新收到。如果网络一直不通，SDK 则会触发 Logout 事件，后台也会将该用户设置为登出状态，并自动踢出频道。

加入频道

- 用户必须登录之后，才能加入频道。
- 用户一次只能加入一个频道，加入频道会自动离开其他频道。
- 用户登出后，自动离开频道。
- 用户加入频道后，可以自动得到所有用户列表

注：大型频道(例如，人数超过 200 人)的用户不会自动得到所有用户列表。

频道事件

- 用户加入频道之后，可以收到所有的频道事件（如其他用户加入离开频道消息）。
- 频道事件也是可靠的。

频道消息

- 频道消息是广播消息，所有频道用户都能收到。

频道属性

- 每个频道有一个属性表。
- 用户加入时，可以获得属性列表。
- 每个用户都可以修改属性。
- 当属性变化时，每个用户都会得到通知。

呼叫系统

概览

呼叫系统用来发起呼叫，邀请其他用户加入某个频道，实现双人或者多人的音视频通话：

- 被叫必须处于登录状态才能收到呼叫。
- 用户可以呼叫 **Agora** 内部用户。内部用户指注册了 **Agora** 信令系统的用户。
- 用户可以通过 **SIP** 网关呼叫外部用户。通过 **SIP** 网关可以对接：**PSTN** 落地电话、呼叫中心、**PBX**、**IM** 等各种外部用户。
- 呼叫一次只能呼叫 1 个人。多人音视频可以发起多次呼叫。每个呼叫是独立的，可以呼叫 **Agora** 内部用户，或者是通过 **SIP** 网关的外部用户（落地电话用户、呼叫中心座席）。
- 呼叫可以通过 **Server API** 发起。当多人音视频通信时，通过服务端发起呼叫，并将呼叫状态通过频道属性或者消息下发给其他用户，是更可靠的方式。

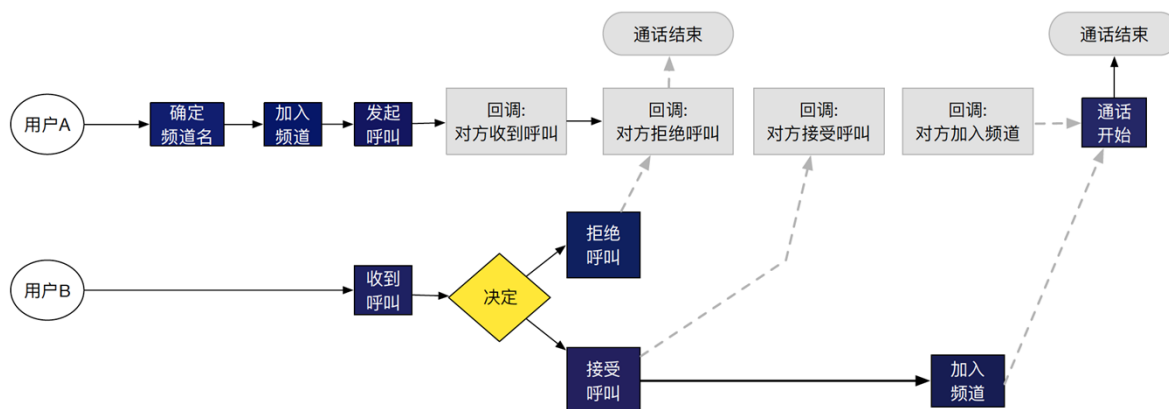
注：不使用呼叫系统也可以实现音视频通信，前提是双方要主动加入同一个频道。

呼叫流程

呼叫和频道是独立的。可以选择以下任一方式：

- 先呼叫，等对方接通之后，大家一起加入频道。
- 先加入频道，然后再呼叫。这样被叫收到邀请，加入频道可以更快地通话。这样的做法也支持被叫在接听之前，先看到频道内的模糊视频等功能。注意：如果被叫不接听，而主叫已加入频道，会有一定费用。统计上，也产生一个单人频道的费用。

以下为先加入频道的呼叫流程：



通过 SIP 网关呼叫外部系统，SIP 指令会被转换为上述指令，另外系统还支持：DTMF，用来模拟电话的按键，适用于呼叫中心、PBX 等。

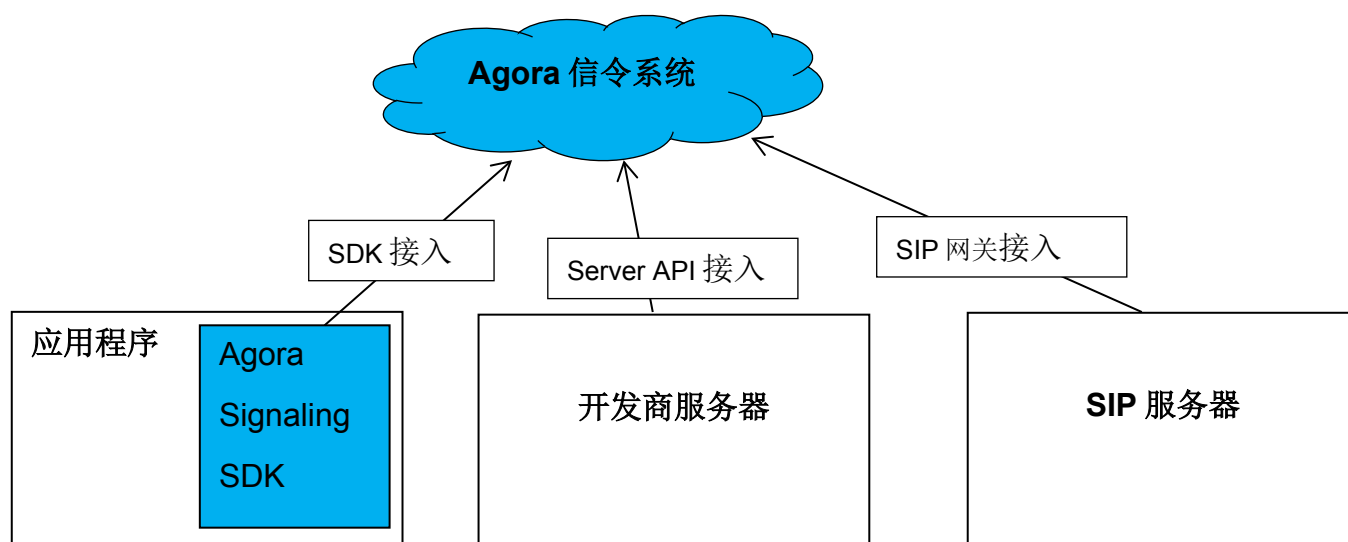
呼叫超时

如果呼叫在 30 秒内没有收到确认，则呼叫失败。

呼叫收到确认后，超时由主叫控制。一定时间对方未接受或拒绝，主叫可以结束呼叫。

接入方式

可以通过以下三种方式接入 Agora 信令系统：



SDK 接入

SDK 接入方式主要支持以下功能：

- 账户消息系统：
 - 用户登录
 - 点对点消息
- 呼叫系统
- 频道系统

更多详细内容，见 [Signaling SDK API 参考](#)。

Server API 接入

Server API 接入方式主要支持以下功能：

- 账号消息系统：
 - Server API：免登录
 - 查阅用户状态
 - 订阅用户状态属性变化
 - 给用户发送消息
- 频道系统：
 - 查询频道成员
- 呼叫系统：
 - 订阅呼叫事件
 - 发起呼叫

更多详细内容，见 **Server API 参考手册**。

SIP 网关接入

SIP 网关用来对接外部系统，包括：

- 信令对接，将私有的信令转为 SIP 协议。
- 媒体对接，将私有的媒体转为 RTP 协议。

目前支持的信令对接功能包括：

- 呼叫功能
- 自定义呼叫参数

注：如需开通 SIP 网关访问 Agora 信令系统，请联系 support@agora.io。

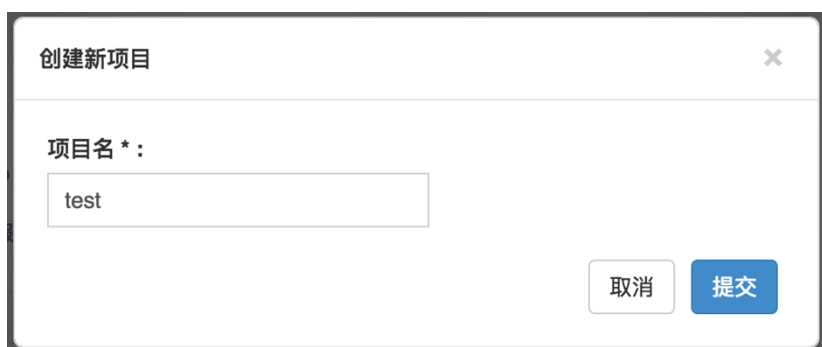
获取和使用 Signaling Key

使用 Agora Signaling SDK 接入 Agora 信令系统的用户都需要提供 Signaling Key，确保用户每次登录都经过信令服务器的验证。Signaling Key 采用 HMAC/SHA1 签名方案，提高了系统及通话的安全性。

获取 App ID 和 App Certificate

每个 Agora 账户下可创建多个项目，且每个项目均可启用独立的 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**。即可查看该项目的 App ID。



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。

- 请将 App Certificate 保存于服务器端，且对任何客户端均不可见。
- App Certificate 使用方法见下文 [Signaling Key 结构](#)。

集成 Signaling Key 算法

使用以下算法生成登录 Agora 信令系统所需的 token(Signaling Key):

输入:

```
appId      = "C5D15F8FD394285DA5227B533302A518" //App ID
appCertificate = "fe1a0437bf217bdd34cd65053fb0fe1d" //App Certificate
expiredTime  = "2592000" // expiredTime
account      = "test@agora.io" //用户登录厂商 app 的账号
```

输出:

```
token

= "1:appId:expiredTime:md5(account + appId + appCertificate + expiredTime)"
= "1:C5D15F8FD394285DA5227B533302A518:2592000:md5(test@agora.ioC5D15F8FD394285DA52
27B533302A518fe1a0437bf217bdd34cd65053fb0fe1d2592000)"
= "1:C5D15F8FD394285DA5227B533302A518:2592000:5c0ee12fdf2020d0d0fdad04d6395473"
```

注: 关于上述字段的解释, 详见 [Signaling Key 结构](#)。

使用 Signaling Key

每一次客户端请求登录 Agora 信令系统时:

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、客户端用户账号, 有效期时间戳等信息通过 Agora 提供的算法生成 Signaling Key, 返回给授权的客户端应用程序。
3. 客户端应用程序调用 login, 参数 token 要求设为 Signaling Key。
4. Agora 的服务器接收到 Signaling Key 信息, 验证该请求是来自于合法用户, 并允许访问 Agora 信令系统。

Signaling Key 的安全性

Signaling Key 结构

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		Signaling Key 算法的版本号, 固定为 1

App ID	字符串	32	appId: 32 位 App ID 字符串
服务到期时间戳	数字	10	expiredTime: 服务到期的 UTC 时间戳，用户在服务到期后，无法再登录 Agora 信令系统和使用其功能。
签名	字符串	32	<p>签名的 hex 编码。根据以下字段的输入，通过 MD5 算法和 hex 编码而成的字符串：</p> <ul style="list-style-type: none"> ✓ account: 用户登录厂商 app 的账号 ✓ appId: 32 位 App ID 字符串 ✓ appCertificate: 32 位 App Certificate 字符串 ✓ expiredTime: 服务到期的 UTC 时间戳，用户在服务到期后，无法再登录 Agora 信令系统和使用其功能。

表 1

集成和初始化 SDK

1. 将 libs/nomedia/下的库添加到客户端。
2. 初始化代码。

```
inst = [AgoraAPI getInstanceWithoutMedia:appId];
instMedia = [AgoraRtcEngineKit sharedEngineWithappId:appId delegate:nil];
```

3. 登陆成功时，保存 uid。

```
inst.onLoginSuccess = ^(uint32_t uid, int fd){
    my_uid = uid;
};
```

4. 用户需同时离开或加入媒体和信令频道。

```
[inst channelJoin:self.txtChannel.text];  
  
[instMedia joinChannelByKey:appld channelName:self.txtChannel.text info:nil uid:my_uid  
joinSuccess:nil];
```

注：详细代码，参见包含在 Agora Signaling SDK 软件包内的示例代码(demo)。

Signaling SDK API 参考

方法

登录(login)

```
(void) login:(NSString*)appId account:(NSString*)account token:(NSString*)token  
uid:(uint32_t)uid deviceId:(NSString*)deviceId ;
```

该方法用于登录 Agora 信令系统。用户在进行任何操作以前，必须先登录。

登录成功会回调 `onLoginSuccess`，登录失败会回调 `onLoginFailed`。如果登录之后失去和服务器的连接，会回调 `onLogout`。

参数名称	描述
appId	厂商的 App ID
account	用户登录厂商 app 的账号，最大 128 可见字符（暂时不能使用空格）。可以是用户的 uid、昵称、guid 等任何内容，但必须保证唯一。本文提到的所有 account 参数都是如此。
token	由 App ID 和 App Certificate 生成的 Signaling Key，详见 获取和使用 Signaling Key 。
uid	（该参数已废弃）固定填 0
deviceId	暂时无用，设置为空

退出(logout)

```
(void) logout;
```

该方法让用户退出当前登录。退出当前登录之后，加入的频道也会自动退出。

加入频道(channelJoin)

```
(void) logout; (void) channelJoin:(NSString*)channelID;
```

该方法让用户加入指定频道。用户一次只能加入一个频道。如加入指定频道时已在其他频道中，将自动从其他频道退出。

用户加入频道成功后，自己将收到回调 `onChannelJoined`，其他同一频道内用户将收到回调 `onChannelUserJoined`。用户加入失败后，自己将收到回调 `onChannelJoinFailed`。

参数名称	描述
channelID	频道名，最大 128 可见字符

离开频道(channelLeave)

(void) channelLeave:(NSString*)channelID;

该方法用于退出当前的频道。退出成功后，所有频道用户将收到回调 onChannelUserLeaved。

参数名称	描述
channelID	频道名，最大 128 可见字符

查询频道用户数(channelQueryUserNum)

(void) channelQueryUserNum:(NSString*)channelID;

该方法用于查询指定频道的用户数量。成功失败都会回调 onChannelQueryUserNumResult。

参数名称	描述
channelID	频道名，最大 128 可见字符

设置频道属性(channelSetAttr)

(void) channelSetAttr:(NSString*)channelID name:(NSString*)name value:(NSString*)value;

该方法用于设置频道属性。当操作成功，所有频道用户都将收到 onChannelAttrUpdated 事件。

参数名称	描述
channelID	频道名，最大 128 可见字符
name	属性名（最大 128 可见字符）
value	属性值（最大 1024 可见字符）

删除频道属性(channelDelAttr)

(void) channelDelAttr:(NSString*)channelID name:(NSString*)name;

该方法用于删除频道属性。

参数名称	描述
channelID	频道名，最大 128 可见字符
name	属性名（最大 128 可见字符）

删除所有频道属性(channelClearAttr)

(void) channelClearAttr:(NSString*)channelID;

该方法用于删除所有频道属性。

参数名称	描述
channelID	频道名，最大 128 可见字符

发起呼叫(channelInviteUser)

(void) channelInviteUser:(NSString*)channelID account:(NSString*)account uid:(uint32_t)uid ;

该方法用于发起呼叫，即邀请某用户加入某个频道。

呼叫和加入频道，是两个独立的过程。用户必须自己再另行加入频道，用户可以选择：

先加入频道，再发送呼叫邀请或先发送呼叫邀请，对方接受后再加入频道。

如果呼叫失败，会回调 onInviteFailed。可能的原因有：

- 对方不在线
- 本端网络不通
- 服务器异常

如果收到对方的确认信息：

本地将回调 onInviteReceivedByPeer, 对方会回调 onInviteReceived。

参数名称	描述
channelID	频道名，最大 128 可见字符
Account	对方登录厂商 app 的账号。
uid	废弃字段，填 0

发起呼叫(channelInviteUser2)

(void) channelInviteUser2:(NSString*)channelID account:(NSString*)account extra:(NSString*)extra;

同上述 API，但参数不同。

参数名称	描述
channelID	频道名，最大 128 可见字符
account	对方登录厂商 app 的账号。

Extra	废弃字段，或者一个 json 字符串
-------	--------------------

向远端发送 DTMF 消息(channelInviteDTMF)

```
(void) channelInviteDTMF:(NSString*)channelID phoneNum:(NSString*)phoneNum
dtmf:(NSString*)dtmf;
```

该方法发送 DTMF 消息到对端，用于 SIP 网关的呼叫。

参数名称	描述
channelID	频道名，最大 128 可见字符
phoneNum	对方的电话号码
dtmf	拨通后，后续需要输入的模拟电话按键(DTMF)

接受呼叫(channelInviteAccept)

```
(void) channelInviteAccept:(NSString*)channelID account:(NSString*)account
uid:(uint32_t)uid ;
```

该方法用于当收到呼叫时，调用本接口接受收到的呼叫。

对方会收到 onInviteAcceptedByPeer 回调。

参数名称	描述
channelID	频道名，最大 128 可见字符
account	对方登录厂商 app 的账号
uid	废弃字段，填 0

拒绝呼叫(channelInviteRefuse)

```
(void) channelInviteRefuse:(NSString*)channelID account:(NSString*)account
uid:(uint32_t)uid ;
```

该方法用于当收到呼叫时，调用本接口拒绝收到的呼叫。

对方会收到 onInviteRefusedByPeer 的回调。

参数名称	描述
channelID	频道名，最大 128 可见字符
account	对方登录厂商 app 的账号
uid	废弃字段，填 0

关闭呼叫(channelInviteEnd)

```
(void) channelInviteEnd:(NSString*)channelID account:(NSString*)account uid:(uint32_t)uid;
```

该方法用于当呼叫接通后，调用本接口关闭呼叫。

本端会回调 onInviteEndByMyself，对端会回调 onInviteEndByPeer。

参数名称	描述
channelID	频道名，最大 128 可见字符
account	对方登录厂商 app 的账号
uid	废弃字段，填 0

发送点对点消息(messageInstantSend)

```
(void) messageInstantSend:(NSString*)account uid:(uint32_t)uid msg:(NSString*)msg msgID:(NSString*)msgID;
```

该方法发送点对点消息到某个指定账号。对方将收到 onMessageInstantReceive 回调。

发送成功本地将回调 onMessageSendSuccess，发送失败将回调 onMessageSendError。

参数名称	描述
account	对方登录厂商 app 的账号。
uid	废弃字段，填 0。
msg	消息正文。 每条消息最大为 8K 可见字符。
msgID	可见字符，可填空，用于回调的消息标示。

发送频道消息(messageChannelSend)

```
(void) messageChannelSend:(NSString*)channelID msg:(NSString*)msg msgID:(NSString*)msgID;
```

该方法用于发送频道消息，频道内所有成员会收到 onMessageChannelReceive 回调。

发送成功会回调 onMessageSendSuccess，发送失败会回调 onMessageSendError。

参数名称	描述
channelID	频道名，最大 128 可见字符
msg	消息正文。 如果是大规模群组通话，每条消息最大为 1K 可见字符。

	如果是一对一通话，每条消息最大为 8K 可见字符。
msgID	可见字符，可填空，用于回调的消息标示。

设置用户属性(setAttr)

(void) setAttr:(NSString*)name value:(NSString*)value;

该方法用于设置用户属性。

参数名称	描述
name	属性名, 最大 256 字节可见字符
Value	属性值, 最大 1K 可见字符

获取自己的用户属性(getAttr)

(void) getAttr:(NSString*)name;

该方法用于获取自己的用户属性。调用成功会回调 onUserAttrResult。

参数名称	描述
name	属性名, 最大 256 字节可见字符

获取自己的所有用户属性(getAttrAll)

(void) getAttrAll;

该方法用于获取自己的所有用户属性。调用成功会回调 onUserAttrResult。

获取某个用户的用户属性(getUserAttr)

(void) getUserAttr:(NSString*)account name:(NSString*)name;

该方法用于获取某个用户的用户属性。调用成功会回调 onUserAttrResult。

参数名称	描述
account	该用户登录厂商 app 的账号。
name	最大 256 字节可见字符

获取某个用户的所有用户属性(getUserAttrAll)

该方法用于获取某个用户的所有用户属性。调用成功会回调 onUserAttrAllResult。

(void) getUserAttrAll:(NSString*)account;

参数名称	描述
------	----

account	该用户登录厂商 app 的账号。
---------	------------------

检查当前是否在线(isOnline)

(bool) isOnline;

该方法用于检查自己是否登录成功, 当前是否在线。

参数名称	描述
Return Value	<ul style="list-style-type: none"> 1: 在线 0: 不在线

内测接口和过期接口

接口	说明	原型
messageAppSend	内测接口	(void) messageAppSend:(NSString*)msg msgID:(NSString*)msgID ;
messageChannelSendFast	内测接口	(void) messageChannelSendFast:(NSString*)channelID msg:(NSString*)msg msgID:(NSString*)msgID ;
messagePushSend	内测接口	(void) messagePushSend:(NSString*)account uid:(uint32_t)uid msg:(NSString*)msg msgID:(NSString*)msgID ;
messageChatSend	内测接口	(void) messageChatSend:(NSString*)account uid:(uint32_t)uid msg:(NSString*)msg msgID:(NSString*)msgID ;
messageDTMFSend	废弃接口	(void) messageDTMFSend:(uint32_t)uid msg:(NSString*)msg msgID:(NSString*)msgID ;
setBackground	内测接口	(void) setBackground:(uint32_t)bOut ;
setNetworkStatus	内测接口	(void) setNetworkStatus:(uint32_t)bOut ;
Ping	内测接口	(void) ping;
channelInvitePhone	废弃接口	(void) channelInvitePhone:(NSString*)channelID phoneNum:(NSString*)phoneNum uid:(uint32_t)uid ;
channelInvitePhone2	废弃接口	(void) channelInvitePhone2:(NSString*)channelID phoneNum:(NSString*)phoneNum sourcesNum:(NSString*)sourcesNum ;
channelInvitePhone3	废弃接口	(void) channelInvitePhone3:(NSString*)channelID phoneNum:(NSString*)phoneNum sourcesNum:(NSString*)sourcesNum extra:(NSString*)extra ;

回调事件

连接丢失回调(**onReconnecting**)

@property (copy) void(^onReconnecting)(uint32_t nretry) ;

登录成功后，丢失连接触发本事件。

参数名称	描述
nretry	当前重连的次数

重连成功回调(**onReconnected**)

@property (copy) void(^onReconnected)(int fd) ;

当重连成功会触发此回调。

参数名称	描述
fd	内部使用

登录成功回调(**onLoginSuccess**)

@property (copy) void(^onLoginSuccess)(uint32_t uid,int fd) ;

当登录成功后触发此回调。

参数名称	描述
uid	废弃
fd	内部使用

退出登录回调(**onLogout**)

@property (copy) void(^onLogout)(AgoraEcode ecode) ;

当退出登录时触发此回调。

参数名称	描述
ecode	错误码

登录失败回调(**onLoginFailed**)

当登录失败时触发此回调。

@property (copy) void(^onLoginFailed)(AgoraEcode ecode) ;

参数名称	描述
ecode	错误码

加入频道回调(**onChannelJoined**)

@property (copy) void(^onChannelJoined)(NSString* channelId) ;

当加入频道成功时触发此回调。

参数名称	描述
channelID	频道名

加入频道失败回调(**onChannelJoinFailed**)

@property (copy) void(^onChannelJoinFailed)(NSString* channelId,AgoraEcode ecode) ;

当加入频道失败触发此回调。

参数名称	描述
channelID	频道名
ecode	错误码

离开频道回调(**onChannelLeaved**)

@property (copy) void(^onChannelLeaved)(NSString* channelId,AgoraEcode ecode) ;

当离开频道成功触发此回调。

参数名称	描述
channelID	频道名
ecode	错误码

其他用户加入频道回调(**onChannelUserJoined**)

@property (copy) void(^onChannelUserJoined)(NSString* account,uint32_t uid) ;

当有用户加入频道触发此回调。

参数名称	描述
account	该用户登录厂商 app 的账号。
uid	废弃字段

其他用户离开频道回调(**onChannelUserLeaved**)

@property (copy) void(^onChannelUserLeaved)(NSString* account,uint32_t uid) ;

当有用户离开频道触发此回调。

参数名称	描述
accounts	该用户登录厂商 app 的账号。
uid	废弃字段

获取频道内用户列表回调(onChannelUserList)

```
@property (copy) void(^onChannelUserList)(NSMutableArray* accounts, NSMutableArray* uids);
```

当加入频道成功后，会触发此回调。

参数名称	描述
accounts	用户账号列表
uids	废弃字段

返回查询的用户数量回调(onChannelQueryUserNumResult)

```
@property (copy) void(^onChannelQueryUserNumResult)(NSString* channelId,AgoraEcode ecode,int num) ;
```

查询频道用户数量触发。

参数名称	描述
channelID	频道名
ecode	错误码
num	查询结果

频道属性发生变化回调(onChannelAttrUpdated)

```
@property (copy) void(^onChannelAttrUpdated)(NSString* channelId,NSString* name,NSString* value,NSString* type) ;
```

当频道属性变化时触发。

参数名称	描述
channelID	频道名
name	属性名
value	属性值
type	变化类型: <ul style="list-style-type: none"> "update" : 更新

	<ul style="list-style-type: none"> • "del" : 删除 • "clear": 全部删除
--	---

收到呼叫邀请回调(**onInviteReceived**)

@property (copy) void(^onInviteReceived)(NSString* channelId,NSString* account,uint32_t uid);

当收到呼叫邀请触发。

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号。
uid	废弃字段

远端已收到呼叫回调(**onInviteReceivedByPeer**)

@property (copy) void(^onInviteReceivedByPeer)(NSString* channelId,NSString* account,uint32_t uid);

当呼叫被对方收到时触发。

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号
uid	废弃字段

远端已接受呼叫回调(**onInviteAcceptedByPeer**)

@property (copy) void(^onInviteAcceptedByPeer)(NSString* channelId,NSString* account,uint32_t uid);

当呼叫被对方接受时触发。

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号
uid	废弃字段

对方已拒绝呼叫回调(**onInviteRefusedByPeer**)

@property (copy) void(^onInviteRefusedByPeer)(NSString* channelId,NSString* account,uint32_t uid) ;

当呼叫被对方拒绝时触发。

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号
uid	废弃字段

呼叫失败回调(**onInviteFailed**)

@property (copy) void(^onInviteFailed)(NSString* channelId,NSString* account,uint32_t uid,AgoraEcode ecode) ;

当呼叫失败时触发。

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号
uid	废弃字段

对方已结束呼叫回调(**onInviteEndByPeer**)

@property (copy) void(^onInviteEndByPeer)(NSString* channelId,NSString* account,uint32_t uid) ;

当呼叫被对方结束时触发。

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号
msg	消息正文

本地已结束呼叫回调(**onInviteEndByMyself**)

当呼叫被自己结束时触发。

@property (copy) void(^onInviteEndByMyself)(NSString* channelId,NSString* account,uint32_t uid) ;

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号
uid	废弃字段
msg	消息正文

消息发送失败回调(onMessageSendError)

@property (copy) void(^onMessageSendError)(NSString* messageID,AgoraEcode ecode) ;

当发送消息失败时触发。

参数名称	描述
messageID	消息 ID
ecode	错误码

消息已发送成功回调(onMessageSendSuccess)

@property (copy) void(^onMessageSendSuccess)(NSString* messageID) ;

当发送消息成功时触发。

参数名称	描述
messageID	消息 ID

收到用户消息回调(onMessageInstantReceive)

@property (copy) void(^onMessageInstantReceive)(NSString* account,uint32_t uid,NSString* msg) ;

当收到用户消息时触发。

参数名称	描述
account	对方登录厂商 app 的账号
uid	废弃字段
msg	消息正文

收到频道消息回调(onMessageChannelReceive)

@property (copy) void(^onMessageChannelReceive)(NSString* channelId,NSString* account,uint32_t uid,NSString* msg) ;

当收到频道消息时触发。

参数名称	描述
channelID	频道名
account	对方登录厂商 app 的账号
uid	废弃字段
msg	消息正文

已打印日志回调(onLog)

@property (copy) void(^onLog)(NSString* txt) ;

当有日志打印时触发。

参数名称	描述
txt	一行日志的内容

已获取用户属性查询结果回调(onUserAttrResult)

@property (copy) void(^onUserAttrResult)(NSString* account,NSString* name,NSString* value) ;

用户属性查询结果回调。

参数名称	描述
account	对方登录厂商 app 的账号
Name	属性名
value	所有属性的 json 值

已获取所有用户属性查询结果回调(onUserAttrAllResult)

@property (copy) void(^onUserAttrAllResult)(NSString* account, NSString* value) ;

用户属性查询结果回调。

参数名称	描述
account	用户登录厂商 app 的账号
value	所有属性的 json 值

内测接口和过期接口

接口	说明	原型
onMessageAppReceived	内测接口	@property (copy) void(^onMessageAppReceived)(NSString* msg) ;

错误代码

错误代码	值	描述
SUCCESS	0	没有错误
LOGOUT_E_OTHER	100	未知原因
LOGOUT_E_USER	101	用户主动退出
LOGOUT_E_NET	102	网络问题
LOGOUT_E_KICKED	103	相同账号在其他地方登录
LOGOUT_E_PACKET	104	已废弃
LOGOUT_E_TOKENEXPIRED	105	Signaling Key 已过期
LOGOUT_E_OLDVERSION	106	已废弃
LOGOUT_E_TOKENWRONG	107	已废弃
LOGIN_E_OTHER	200	未知原因
LOGIN_E_NET	201	网络问题
LOGIN_E_FAILED	202	服务器拒绝
LOGIN_E_CANCEL	203	登录被用户取消
LOGIN_E_TOKENEXPIRED	204	Signaling Key 过期，拒绝登录
LOGIN_E_OLDVERSION	205	已废弃
LOGIN_E_TOKENWRONG	206	Signaling Key 异常
LOGIN_E_TOKEN_KICKED	207	用户使用更新的 Signaling Key 在其他地方登录过了
JOINCHANNEL_E_OTHER	300	加入频道失败
SENDMESSAGE_E_OTHER	400	发送消息失败
SENDMESSAGE_E_TIMEOUT	401	发送消息超时
QUERYUSERNUM_E_OTHER	500	查询频道用户数量失败
QUERYUSERNUM_E_TIMEOUT	501	查询频道用户数量超时
QUERYUSERNUM_E_BYUSER	501	已废弃
LEAVECHANNEL_E_OTHER	600	未知原因离开频道
LEAVECHANNEL_E_KICKED	601	被管理员踢出频道
LEAVECHANNEL_E_BYUSER	602	用户自己离开频道

LEAVECHANNEL_E_LOGOUT	603	logout 时被踢出频道
LEAVECHANNEL_E_DISCONN	604	断线时离开频道
INVITE_E_OTHER	700	呼叫失败
INVITE_E_REINVITE	701	重复呼叫
INVITE_E_NET	702	网络问题
INVITE_E_PEEROFFLINE	703	对方不在线
INVITE_E_TIMEOUT	704	呼叫超时
INVITE_E_CANTRECV	705	已废弃
GENERAL_E	1000	通用错误
GENERAL_E_FAILED	1001	通用错误—失败
GENERAL_E_UNKNOWN	1002	通用错误—未知
GENERAL_E_NOT_LOGIN	1003	通用错误—未登录时进行操作
GENERAL_E_WRONG_PARAM	1004	通用错误—调用参数错误