Agora Native SDK for iOS

Reference Manual v1.4

support@agora.io

June 2016

Content

Introduction

Agora CaaS

Agora Communications as a Service (CaaS) provides ensured Quality of Experience for worldwide, Internet-based voice and video communications through the Agora Global Network. The network optimizes real-time, mobile communications and solves quality of experience challenges for mobile devices, such as 3G/4G/Wi-Fi networks that perform erratically and Internet bottlenecks worldwide.

Agora CaaS includes mobile-optimized Agora Native SDKs for iOS and Android smartphones that provide access to the Agora Global Network along with device-specific mobile optimizations. Agora Native SDK is also available for Windows and Mac. Applications using the Native SDK link it directly into the application when they are built. Agora Native SDK is also available for Windows and Mac. Applications using the Native SDK link it directly into the application when they are built. See ***Getting Started*** for details on how to deploy and use the Agora Native SDK for iOS.

The Agora Native SDK for Web is a further capability to the Agora CaaS capabilities. The SDK provides web browsers with open access to the Agora Global Network.

Agora Native SDK for iOS

The **Agora Native SDK for iOS** allows your code to perform the following operations:

- **Session setup**: join and leave shared Agora.io conferencing sessions (identified by unique channel names) where there may be many global users in one conference or simply one other user that needs one-to-one communication. Unique channel names should be created and managed by your application code and are usually constructed from user, session and date/time information your application is already managing.
- **Media control**: enable and disable voice and video (allowing muting) and set various voice and video parameters that help the Agora SDK optimize communications. Many of the SDK operations are automated and do not require developer intervention if these parameter settings are not provided.
- **Device control**: access the microphone or speakerphone, set the volume, select from alternative cameras and set the video window display.
- **Session control**: receive events when other users join or leave a conference session (channel), and understand who's speaking, who's muted, and who's viewing the session. These events allow the application to decide when to make changes to the delivery of video and audio streams, and other application-specific user and status information.
- **Quality management**: obtain statistics and quality indicators about network conditions, run built-in tests, submit ratings and complaints about sessions and enable different levels of error logging.
- **Recording**: record audio or video and audio in one or multiple channels simultaneously. This function is only applicable to the users who have adopted the Dynamic Key schema. For how to deploy and use the recording function, refer to the document ***Agora Recording Server User Guide***.

All documents and full SDKs can be downloaded at www.agora.io/developer.

The Agora Native SDK for iOS provides multiple Object-C classes to deliver these features.

- The `AgoraRtcEngineKit` class provides all the methods that can be invoked by your application.
- The `AgoraRtcEngineDelegate` class uses delegate methods to provide event notifications to your application.

Delegate methods are replacing the use of some Block callbacks from version 1.1 of the SDK. The Block callbacks are also documented below. However, where appropriate we recommend replacing them with Delegate methods. The SDK calls the Block method if a callback is defined in both Block and Delegate.

For detailed API usage, see *Agora Native SDK for iOS API Reference*.

Getting Started

Obtaining the SDK

The **Agora Native SDK for iOS** is available from agora.io/developer or contact sales@agora.io to get access to the latest version of our SDK.

Obtaining a Vendor Key

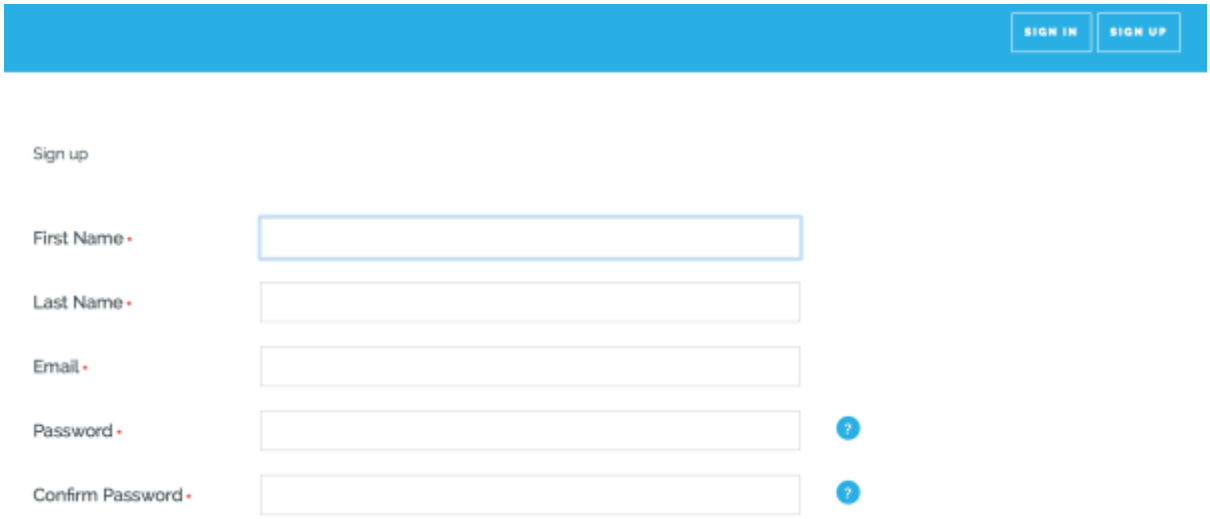
Developers must obtain a **vendor key.** The key is required when you use APIs to access the Agora Global Network.

This section only describes how to use the Vendor Key directly, but you can also combine it with a sign key within your web server code to generate a more secure dynamic key:

For why and how to use dynamic keys (**recommended**), see *Using Dynamic Keys for Increased Security*.

To obtain your vendor key:

1. Sign up for a new account at https://dashboard.agora.io.



2. Complete the sign-up process.

   Once the new account is created, you will find your API vendor key on the dashboard. You will also receive an email with your API vendor key.

3. Once the **vendor key** is assigned, you can start using Agora.io services with this key.

Requirements

Required Development Environment

- Apple XCode version 6.0 or higher
- iOS simulator or real devices with audio functionality
  Note: Video functionality is only supported on real devices.

Required Libraries

Agora Native SDK for iOS requires iOS 6.0 SDK or a later version. Make sure that your main project is linked to the following libraries in the SDK:

- AgoraRtcEngineKit.framework (the Agora Native SDK)
- AudioToolbox.framework
- VideoToolbox framework
- AVFoundation.framework
- CoreMedia.framework
- CoreTelephony.framework

- CoreMotion.framework
- SystemConfiguration.framework
- libc++.dylib

Note: By default, **Agora Native SDK** uses libc++ (LLVM). Contact support@agora.io if you prefer to use libstdc++ (GNU).

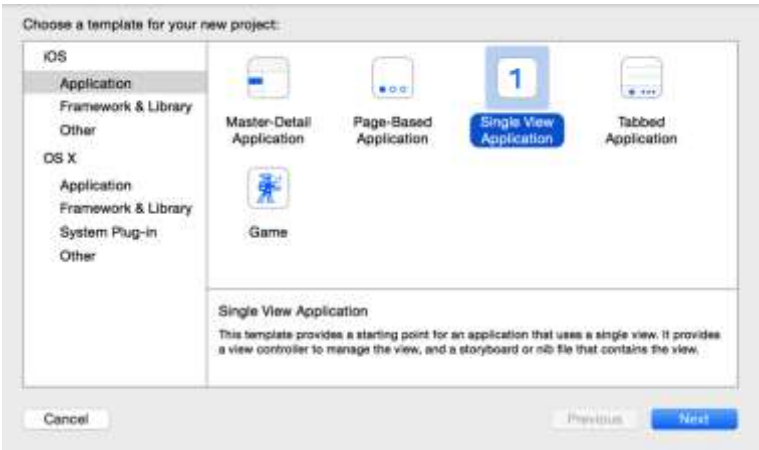
In the source file, use the following command to import AgoraRtcEngineKit:

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

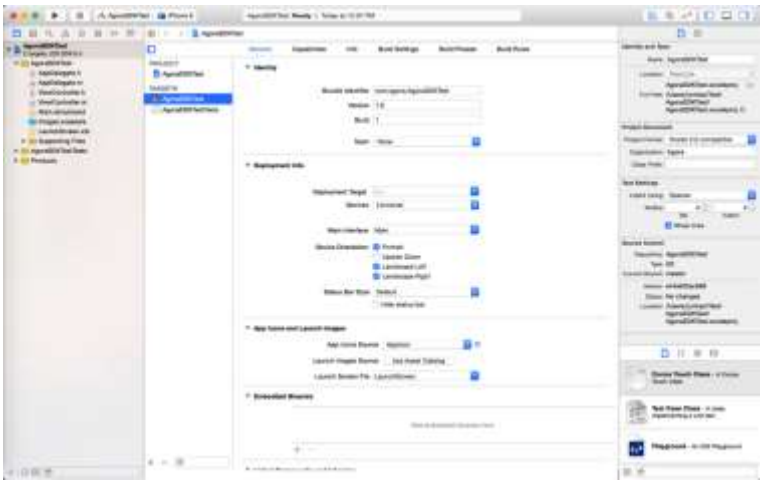
The SDK provides FAT Image libraries with multi-architecture support for both 32/64 bit simulators and 32/64 bit real devices.

Creating a New Project

1. Create a new project in XCode.



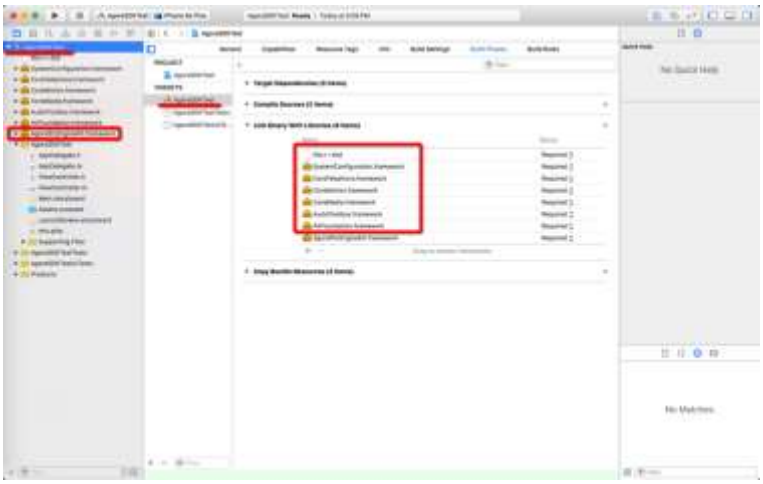
After a project is created, the development environment interface is displayed as below:



2. Navigate to **libs\AgoraRtcEngineKit.framework** in **Agora Native SDK for iOS**.

3. Right click the file **AgoraRtcEngineKit.framework** and select **Add Files to Project** from the pop-up menu to add **AgoraRtcEngineKit.framework** to the project.

4. **AgoraRtcEngineKit.framework** requires some system framework support as listed above under **Required Libraries**. Add the system framework with the **Link Binary with Libraries** in the Build Phase.



5. Include the header file into your source code with #import <AgoraRtcEngineKit/AgoraRtcEngineKit.h> to enable the use of **Agora Native SDK for iOS**.

Refer to the **API Reference** section below for a complete reference of the SDK methods.

Using the Demo Application

Agora provides the following demo application in the **Agora Native SDK for iOS** zip file (describing here the FULL package with both video and voice):

AgoraDemo – Includes basic methods of entering and leaving a call, demonstrates the core operations of **Agora Native SDK for iOS** and enables audio and video calls with simple method calls.

AgoraDemo/AgoraDemo: The source file folder of the demo program
AgoraDemo/AgoraDemo.xcodeproj: The project file of the demo program
AgoraDemo/AgoraDemoTests: The test file folder of the demo program
libs/AgoraRtcEngineKit.framework: The SDK framework

Note: Visit https://github.com/AgoraLab/ for more open source demo applications.

Compiling the Demo Program

1. Run the XCode.
2. Open the **AgoraDemo.xcodeproj** file.

3. Open the project file.

   The Navigation panel on the left displays the file structure of AgoraDemo highlighted as follows:

4. Select the project as shown in Figure 1 callout #1 and then click on **Build and Run** button as shown in callout #2 to start compiling.

   Note:

   - AgoraDemo uses both video and voice and only supports real devices. It does not support simulators.
   - If a “Failed to code sign” error occurs as shown in the callout #3, change the code signing to your activated device, or use your Apple developer account.

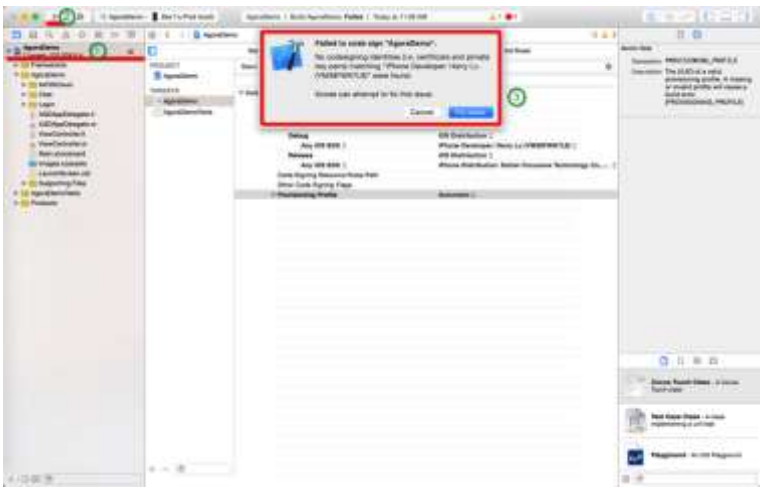


Figure 1

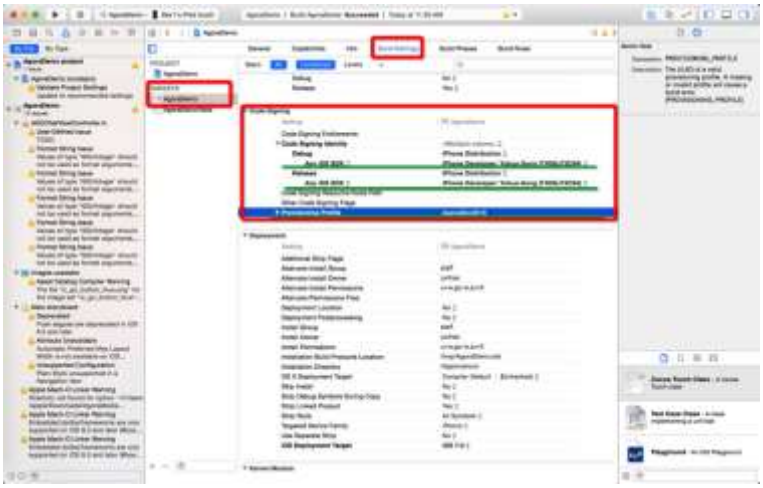


Figure 2

Executing the Demo Program

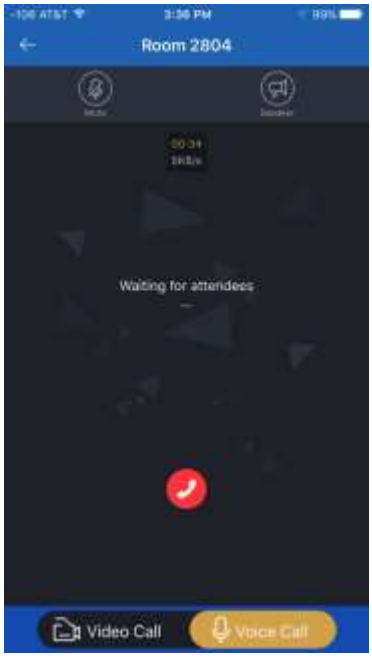
1. Run the AgoraDemo demo program after the deployment to your iOS device is complete.
   The following shows the first page of the demo application, and you need two phone devices to run the demo for both Voice and Video calls:

2. Make an voice call:

   a) On two mobile phones, enter your same **vendor key** and use the same meeting room name (channel name), for example, 2804 as shown below:

   b) Click or tap the **Join Voice Call** button to enter the audio call page.

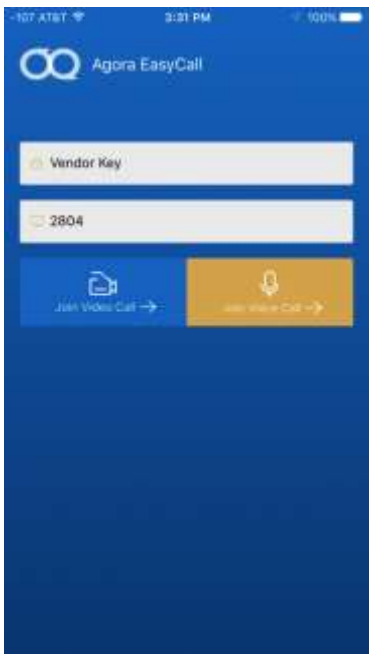
   c) The call page features the following functions: mute/unmute, enable/disable speaker, leave the room.



3. Make a video call:

   a) Click or tap on the **Video Call** button to enter the main call page.

   b) On two mobile phones, enter the same vendor key and use the same meeting room name (channel name).



   c) Click **or tap Join Video Call** to enter the video call page.

   d) The call page features the following functions: mute/unmute, enable/disable speaker, open/close camera, switch cameras, and leave the room.

4. Switch between Video and Voice calls if necessary. Use the **Video Call** and **Voice Call** buttons at the bottom of the screen as shown in the above figure.

   Note: The SDK and the AgoreDemo application support up to 5-way group video sessions, as shown in the following figure. Multiple users join using the same vendor key and room number (channel name in API).

Using Dynamic Keys for Increased Security

Each organization using the Agora SDK has a unique **Vendor Key** for identification. Communications in the Agora Global Network are isolated according to Vendor Keys.

People with different Vendor Keys do not interfere with each other, even when the names of the channels they join are the same.

However, if someone dishonestly obtains your static Vendor Key, then they can use it on their own Agora SDK client applications. If they discover the channel names of your organization, they can even interfere with your communication sessions. To increase the security of applications, we recommend that you implement **Dynamic Keys** in your large-scale production applications.

What Is a Dynamic Key?

A Dynamic Key offers enhanced user authentication for the Agora SDK. Whenever a user tries to enter a channel to access the Agora service, the back-end services of your application use the Vendor Key and a Sign Key provided by Agora to generate a new dynamic key based on keyed-hash message authentication code (HMAC) encryption. The dynamic key is then passed to the client application, which calls the joinChannel API interface function and passes the encoded dynamic key to the Agora server for the user authentication.

Note

- **Vendor Key**：The vendor ID used to identify your organization or product. If dynamic key scheme is not implemented, the Vendor Key is used to access the Agora service.
- **Sign Key**：The “sign” key used to generate the dynamic key (see Table 1 for its usage). The Sign Key is stored on the server and invisible to any users.

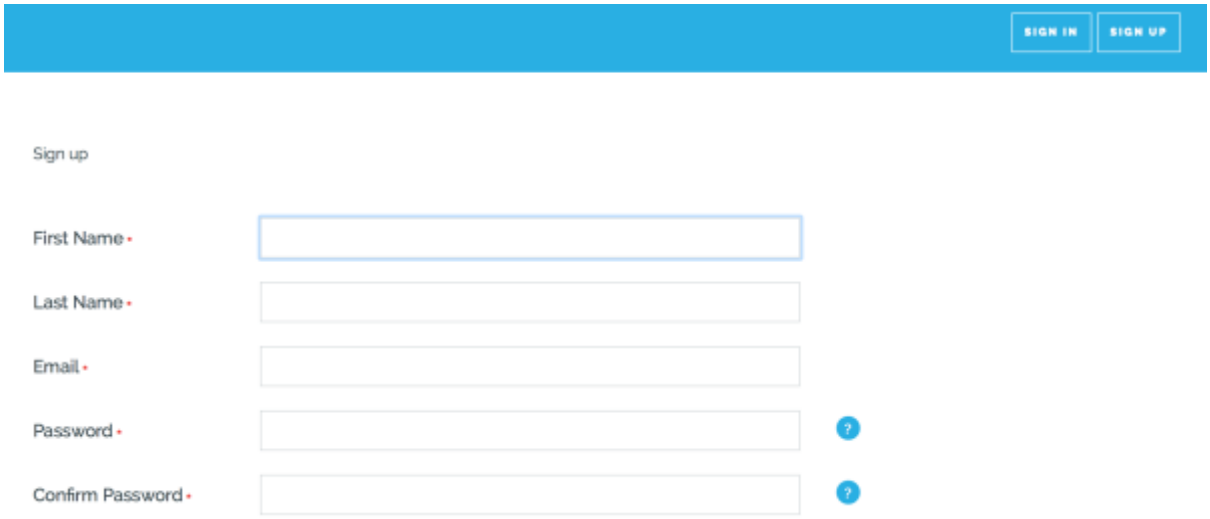
How to Use a Dynamic Key

The following sections describe the use of dynamic keys:

- Obtaining a Vendor Key
- Obtaining a Sign Key
- Implementing the Dynamic Key Scheme
- Using a Dynamic Key

Obtaining a Vendor Key

1. Sign up for a new account at https://dashboard.agora.io.



2. Complete the sign-up process.

   Once the new account is created, find your API vendor key on the dashboard. You also receive an email with your API vendor key.

3. Once a **vendor key** is assigned, you can start using Agora.io services with this key.

Obtaining a Sign Key

1. Generate the Sign Key by clicking **Enable** at https://dashboard.agora.io.

   Caution: Keep your Sign Key on the server and never on any client machine.

2. If you want to renew your Sign Key for some reason, contact support@agora.io.

Implementing the Dynamic Key Scheme

Integrate the dynamic key scheme into your organization's signaling service. Agora.io provides sample server-side code covering the following languages: Java, C++, Python, node.js, and so on, which you can use this code directly in your application. When joining the channel or using the recording function, you will use the same Vendor Key and Sign Key, but the method to generate the Dynamic Key is different. Use generateMediaChannelKey to join the channel, and use generateRecordingKey to use the recording function. For the sample code, refer to any of the following sites:

- Github: https://github.com/AgoraLab/AgoraDynamicKey
- Sign Key Page at dashboard: https://dashboard.agora.io/signkey

Using a Dynamic Key

Before a user joins a channel (that is, has started a call or received a meeting invitation), the following sequence occurs：

1. The client application requests authentication from the signaling server of your organization.
2. The server, upon receiving the request, uses the algorithm provided by Agora.io to generate a dynamic key and then passes the dynamic key back down to the client application.

   The dynamic key is based on the Sign Key, Vendor Key, Channel Name, Current Timestamp, Client User ID, Lifespan Timestamp, and so on.
3. The client application calls the joinChannel() method, which requires the dynamic key as the first parameter.
4. The Agora server receives the dynamic key and confirms that the call comes from a legal user, and then allows it to access the Agora Global Network.

   Note: With the dynamic key scheme implemented, the dynamic key replaces the original Vendor Key when the user joins a channel.

Dynamic keys expire after a certain amount of time and your application must call renewChannelDynamicKey() when a time-out occurs. The onError callback returns ERR_DYNAMIC_KEY_TIMEOUT（109）.

Increased Security with Dynamic Keys

If your organization chooses dynamic keys, before a user joins a channel, the client application must provide a new dynamic key to ensure that every call of the user is authorized by the signaling server. The dynamic key uses the HMAC/SHA1 signature schema to increase security for communications within your organization.

Dynamic Key Structure

Connect all fields in the sequence below (103 bytes in total):

Field	Type	Length	Description
Version	String	3	Dynamic key version information of the algorithm
Sign	String	40	Hex code for the signature. It is a 40-byte string represented by hex code and calculated by HMAC algorithm based on inputs including the Sign Key and the following fields: ● **Service Type**: The visible ASCII string provided by Agora.io. See ***Service Type***. ● **Vendor Key**: The 32-bit vendor key string. ● **Timestamp**: The timestamp created when the dynamic key is generated. ● **Random Number**: A 32-bit integer in hex code; generated upon each query. ● **Channel**: The channel name specified by the user with a maximum length of 64 bytes. ● **User ID**: The User ID defined by the client. ● **Service Expiration Timestamp**: The timestamp indicates that from the specific moment the user cannot use the Agora service any more.
Vendor Key	String	32	Vendor Key
Authorized Timestamp	Number	10	The timestamp represented by the number of seconds elapsed since 1-1-1970. By default, a user can access the Agora service for 5 minutes.
Random Number	Integer	8	A 32-bit integer in hex code; generated upon each query.
Service Expiration	Number	10	This timestamp indicates when the user cannot use the Agora service any more (for example,

Field	Type	Length	Description
Timestamp			the user must leave an ongoing call). Set the value to 0 if no there is no time limit.

Table 1

Service Type

Service	Value	Description
Session	ACS	The audio and video services provided by Agora.io. For example, when calling the joinChannel API, if a Dynamic Key is required, use this service type to generate the Dynamic Key.
Recording	ARS	The audio and video recording services provided by Agora.io. For example, when calling the startRecordingService API, use this service type to generate the Dynamic Key.

Sign Encryption Algorithm: HMAC/SHA1

The dynamic key encoding uses the industry-standard HMAC/SHA1 approach for which there are libraries on most common server-side development platforms, such as Node.js, Java, Python, C++, and so on. For more information, refer to:

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

Contact Agora.io for more information or technical support.

Agora Native SDK for iOS API Reference

AgoraRtcEngineKit Interface Class

AgoraRtcEngineKit is the basic interface class of **Agora Native SDK**. Creating an AgoraRtcEngineKit instance and then calling the methods of this instance enables the use of **Agora Native SDK**'s communication functionality. In previous versions this class was named differently, and it is now renamed to AgoraRtcEngineKit from version 1.0 AgoraAudioKit.

Initialize (sharedEngineWithVendorKey)

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

This method initializes the AgoraRtcEngineKit class as a singleton instance.

Call this method to initialize service before using AgoraRtcEngineKit.

The SDK uses Delegate to inform the application on the engine runtime events. All methods defined in Delegate are optional implementation methods.

Name	Description
vendorKey	The vendor key issued to the application developers by Agora. Apply for a new one from Agora if the key is missing in your kit.
delegate	

Enable Video Mode (enableVideo)

```
(int)enableVideo
```

This method enables video mode.

The application can call this method either before entering a channel or during a call. If it is called before entering the channel, the service starts in video mode; if it is called during a call, it switches from audio to video mode. To disable video mode, call the disableVideo method.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Enable Audio Mode (disableVideo)

```
int disableVideo()
```

This method disables video and enables audio mode.

The application can call this method either before entering a channel or during a call. If it is called before entering the channel, the service starts in audio mode; if it is called during a call, it switches from video to audio mode. To enable video mode, call the enableVideo method.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Start Video Preview (startPreview)

```
(int)startPreview
```

This method starts the local video preview. Before starting the preview, always call setupLocalVideo to setup the preview window and configure its attributes and also call the enableVideo method to enable video. If before calling joinChannelByKey to join the channel, you have called startPreview to start the local video preview, then the local preview is still in the started state after you call leaveChannel to exit the channel. You can call stopPreview to close the local preview.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Stop Video Preview (stopPreview)

```
(int)stopPreview
```

This method stops the local video preview and closes the video.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Join Channel (joinChannelByKey)

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

This method lets the user join a channel.

Users in the same channel can talk to each other; and multiple users in the same channel can start a group chat. Users using different vendor keys cannot call each other. Once in a call, the user must call the leaveChannel method to exit the current call before entering another channel. This method is called asynchronously; therefore, it can be called in the main user interface thread.

Note: The SDK uses iOS's AVAudioSession shared object for audio recording and playing, and so operating on this object may affect the SDK's audio functions. When joining a channel, the SDK calls setCategory AVAudioSessionCategoryPlayAndRecord to set AVAudioSession to PlayAndRecord mode. The application should not set it to any other mode. When setting this mode, the sound being played (for example, a ring tone), if any, is interrupted.

Name	Description
key	The token is a dynamic key generated by the application. This parameter is optional if the user uses a static vendor key. In this case, pass **NULL** as the parameter value. If the user uses a dynamic key, **Agora.io** issues an additional Sign Key to the application developers. The developers can then generate a user key using algorithm and Sign Key provided by **Agora.io** for user authentication on the server. See the discussion above in the ***Getting Started*** section on dynamic keys. In most developer circumstances, the static vendor key should suffice. For users who have a high security requirement, use the Dynamic Key. For users who want to use the recording function, Dynamic Key is mandatory.
channelName	A string providing the unique channel name for the AgoraRTC session. The length must be within 64 bytes. The following is the supported scope: a-z A-Z 0-9 space ! #$%& ()+, -

	:;<=. >? @[] ^_` {|} ~
info	(Optional) Any additional information the developer wants to add. It can be set as NULL Sting or channel related information. Other users in the channel won't receive this information.
uid	Optional) The user ID: a 32-bit unsigned integer ranges from 1 to (2^32-1). It must be unique. If not assigned (or set to 0), the SDK will allocate one and returns it in `joinSuccessBlock` callback. The App needs to record and maintain the returned value as the SDK does not maintain it.

Leave Channel (leaveChannel)

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

This method lets the user leave a channel by hanging up or exiting a call.

After joining a channel, the user must call the leaveChannel method to end the call before joining another one. Calling the leaveChannel method even when not in a call is harmless. The leaveChannel method releases all recourses related to the call. The leaveChannel method is called synchronously, and the user actually does not exit the channel when the call returns. Once the user exited the channel, the SDK triggers didLeaveChannelWithstats callback.

Name	Description
leaveChannelBlock	The callback on user successfully left the channel.
Return Value	0：Method call succeeded. <0：Method call failed.

Renew Channel Dynamic Key (renewChannelDynamicKey)

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

This method updates dynamic key. The key expires after a certain period of time once the Dynamic Key scheme is enabled. When the onError callback reports the error ERR_DYNAMIC_KEY_TIMEOUT(109),the application should retrieve a new token and

then call this method to renew it. Failure to do so will result in SDK disconnecting with the server.

Name	Description
key	The Dynamic Key to be renewed.
Return Value	0：Method call succeeded. <0：Method call failed.

Retrieve Current Call ID (getCallId)

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

Every time a user joins a channel on a client machine by calling `joinChannelByKey`, a CallId is generated to identify the call from this client. Some methods such as rate and complain submit feedback to the SDK and are invoked after the call ends. These methods require assigned values of the CallId parameters. To use these feedback methods, call the getCallId method to retrieve the CallId during the call, and then pass the value as an argument in the feedback methods after the call ends.

Name	Description
Return Value	The current call ID.

Rate the Call (rate)

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

This method lets the user rate the call. It is usually called after the call ends.

Name	Description
callId	The call ID retrieved from the getCallId method.
rating	The rating for the call between 1 (lowest score) to 10 (highest score).
description	A given description for the call with a length less than 800 bytes. This parameter is optional.
Return Value	**0**：Method call succeeded. **<0**：Method call failed. **ERR_INVALID_ARGUMENT (-2)**：The passed argument is invalid, for example, callId invalid. **ERR_NOT_READY (-3)**：The SDK status is incorrect, for example, initialization failed.

Complain about Call Quality (complain)

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

This method allows the user to complain about the call quality. It is usually called after the call ends.

Name	Description
callId	The call ID retrieved from the getCallId method.
description	A given description for the call with a length less than 800 bytes. This parameter is optional.
Return Value	**0**：Method call succeeded. **<0**：Method call failed. **ERR_INVALID_ARGUMENT (-2)**： The passed argument is invalid, for example, callId invalid. **ERR_NOT_READY (-3)**： The SDK status is incorrect, for example, initialization failed.

Start Audio Call Test (startEchoTest)

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

This method launches an audio call test to check if the audio devices (for example, headset and speaker) and the network connection work properly. In the test, the user speaks first, and the recording will be played back in 10 seconds. If the user can hear what he said in 10 seconds, it indicates that the audio devices and network connection work properly.

Note: After calling the startEchoTest method, always call stopEchoTest to end the test, otherwise the application will not be able to run the next echo test, nor can it call the joinChannel method to start a new call.

Name	Description
successBlock	The callback on successfully starting the echo test. See joinSuccessBlock in joinChannelByKey for a description of the callback parameters.
Return Value	**0**：Method call succeeded. **<0**：Method call failed. **ERR_REFUSED (-5)**：Failed to launch the echo test, for example, initialization failed.

Stop Audio Call Test (stopEchoTest)

```
(int)stopEchoTest;
```

This method stops the audio call test.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed. **ERR_REFUSED(-5)**：Failed to stop the echo test. It could be that the echo test is not running.

Enable Network Test (enableNetworkTest)

```
(int)enableNetworkTest
```

This method enables the network quality test to test the user's network connection quality. When enabled, the SDK uses the networkQualityBlock method to notify the application about the network connection quality.

Note: Once the network test is enabled, it uses the network bandwidth even when the application is not in a call. We recommend the following practices:

When the application is in the front end, call the enableNetworkTest method to enable the network connection test; and when the application is switched to the back end, call the disableNetworkTest method to disable the network test in order to reduce network traffic. By default, the network test is disabled.

Name	Description
Return Value	**0**：Method call succeeded.

	<0： Method call failed.

Disable Network Test (disableNetworkTest)

```
(int)disableNetworkTest;
```

This method disables the network connection quality test.

Name	Description
Return Value	**0：** Method call succeeded. **<0：** Method call failed.

Generate Call Quality Report URL (makeQualityReportUrl)

```
(NSString*) makeQualityReportUrl:(NSString*) channel
listenerUid:(NSUInteger) listenerUid
speakerrUid:(NSUInteger) speakerUid
reportFormat:(AgoraRtcQualityReportFormat) reportFormat;
```

This method generates a URL pointing to the call quality reports. The application then uses the returned URL to retrieve detailed call quality data. The application needs to know the channel name and both of the callers' IDs to use this method. For instance, if A and B are in the same channel, the application retrieves a report on A speaking and B listening, and also a report on B speaking and A listening.

Name	Description
channel	The channel name specified in the joinChannel method.
listenerUid	The user ID of the listener.
speakerUid	The user ID of the speaker.
reportFormat	The format of the report. **AgoraRtc_QualityReportFormat_Json (0): JSON.**: Returns the quality report data. The application displays this information to users upon request. **AgoraRtc_QualityReportFormat_Html (1): HTML.**: Returns a report in HTML format, displayed on a web browser or WebVIEW components.
Return Value	The generated URL.

Mute Local Audio Stream (muteLocalAudioStream)

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

This method mutes/unmutes local audio.

Name	Description
mute	Yes: Mutes local audio. No: Unmutes local audio.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Mute All Remote Audio Streams (muteAllRemoteAudioStreams)

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

This method mutes/unmutes all remote callers' audio streams.

Name	Description
muted	Yes: Stops playing all the received audio streams. No: Resumes playing all the received audio streams.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Mute Certain Remote Audio Stream (muteRemoteAudioStream)

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

Mute/unmute a specified remote user's audio stream.
Note: When set to Yes, this method stops playing audio streams without affecting the audio stream receiving process.

Name	Description
uid	The user ID whose audio streams the user intends to mute.
mute	Yes: Stops playing a specified user's audio streams. No: Allows playing a specified user's audio streams.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Enable Speakerphone (setEnableSpeakerphone)

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

This method enables audio output to the speaker.

Note: The SDK calls setCategory AVAudioSessionCategoryPlayAndRecord withOptions to configure the headset/speaker, and so any soundis interrupted when calling this method.

Name	Description
enableSpeaker	Yes： Switches to speaker. No： Switches to headset.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Is Speakerphone Enabled? (isSpeakerphoneEnabled)

```
(BOOL)isSpeakerphoneEnabled;
```

This method tests whether the speakerphone is enabled or not.

Name	Description
Return Value	Yes：Yes, enabled. No：No, not enabled.

Enable Audio Volume Indication (enableAudioVolumeIndication)

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

This method enables the SDK to regularly report to the application on who is speaking and the volume of the speaker.

Name	Description
interval	Specifies the time interval between two consecutive volume indications. **<=0** ： Disables volume indication. **>0** ： The volume indication interval in milliseconds. We recommend setting it to a minimum of 200ms.
smooth	The smoothing factor. Recommended default value is 3.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Set Video Profile (setVideoProfile)

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

This method sets the video encoding profile. Each profile includes a set of parameters such as resolution, frame rate, bitrate, and son.

Note: Always set the video profile before calling the joinChannel method. When the device camera does not support the specified resolution, the SDK automatically chooses a

suitable camera resolution, but the encoder resolution still uses the one specified by setVideoProfile.

Name	Description
profile	The video profile. See AgoraRtcVideoProfile definition for more details.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Video Profile Definition

Video Profile	Enum value	Resolution (width * height)	Frame Rate (fps)	Bitrate (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

Video Profile	Enum value	Resolution (width * height)	Frame Rate (fps)	Bitrate (kbps)
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

Achieving 720P resolution depends on the performance and capabilities of the device and may not currently be possible on lower performance devices. If a device has inadequate performance, low frame rates may result when using 720P. Agora.io is continuing to optimize video for lower-end devices in future releases – contact support@agora.io to discuss particular device needs.

Set Local Video View (setupLocalVideo)

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

This method configures the video display settings on local machine. The application calls this method to bind with the video window (view) of local video streams and configures the video display settings. In the application development, call this method after initialization to configure local video display settings before entering a channel. The bind is still valid after exiting the channel. To unbind the view, set the view value to NULL when calling setupLocalVideo.

Name	Description
local	Configures the video settings. AgoraRTCVideoCanvas types: ● view: The video display window (view). The SDK does not maintain the lifecycle of the view, and it is the responsibility of the application to ensure that

	the view is valid during the call, otherwise it may cause the SDK to crash. The view can be safely released after calling leaveChannel with a returned value. ● renderMode: The video display mode. ● AgoraRtc_Render_Hidden(1): If the video size is different than that of the display window, crops the borders of the video (if the video is bigger) or stretch the video (if the video is smaller) to fit it in the window. ● AgoraRtc_Render_Fit(2): If the video size is different than that of the display window, resizes the video proportionally to fit the window. ● AgoraRtc_Render_Adaptive(3)：If both callers use the same screen orientation, that is, both use vertical screens or both use horizontal screens, the AgoraRtc_Render_Hidden mode applies; if they use different screen orientations, that is, one vertical and one horizontal, the AgoraRtc_Render_Fit mode applies. ● uid: The local user ID as set in the `joinchannel` method.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Set Remote Video View (setupRemoteVideo)

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

This method binds the remote user to the video display window, that is, sets the view for the user of the specified uid. Usually the application should specify the uid of the remote video in the method call before the user enters a channel. If the remote uid is unknown to the application, set it later when the application receives the onUserJoined event. If the Video Recording function is enabled, the Video Recording Service joins the channel as a dumb client, which means others clients will also receive its didJoinedOfUid event. Your application should not bind it with the view, because it does not send any video stream. If your application cannot recognize the dumb client, bind it with the view when the firstRemoteVideoDecodedOfUid. To unbind the user from the view, set the view to null. After the user left the channel, the SDK unbinds the remote user.

Name	Description
remote	Configures the video settings. AgoraRTCVideoCanvas types: • view: The video display window (view). The SDK does not maintain the lifecycle of the view, and it is the responsibility of the application to ensure that the view is valid during the call, otherwise it may cause the SDK to crash. The view can be safely released after calling leaveChannel with a returned value. • renderMode: The video display mode. AgoraRtc_Render_Hidden(1): If the video size is different than that of the display window, crops the borders of the video (if the video is bigger) or stretch the video (if the video is smaller) to fit it in the window. AgoraRtc_Render_Fit(2): If the video size is different than that of the display window, resizes the video proportionally to fit the window. ● AgoraRtc_Render_Adaptive(3)：If both callers use the same screen orientation, that is, both use vertical screens or both use horizontal screens, the AgoraRtc_Render_Hidden mode applies; if they use different screen orientations, that is, one vertical and one horizontal, the AgoraRtc_Render_Fit mode applies. uid：The user ID of the remote user whose video streams are received.

Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Set Local Video Display Mode (setLocalRenderMode)

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

This method configures the local video display mode.
The application can call this method multiple times to change the display mode.

Name	Description
mode	Configures the video mode. AgoraRtcRenderMode types: ● AgoraRtc_Render_Hidden(1): If the video size is different than that of the display window, crops the borders of the video (if the video is bigger) or stretch the video (if the video is smaller) to fit it in the window. ● AgoraRtc_Render_Fit(2): If the video size is different than that of the display window, resizes the video proportionally to fit the window. ● AgoraRtc_Render_Adaptive(3)：If both callers use the same screen orientation, that is, both use vertical screens or both use horizontal screens, the AgoraRtc_Render_Hidden mode applies; if they use different screen orientations, that is, one vertical and one horizontal, the AgoraRtc_Render_Fit mode applies.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Set Remote Video Display Mode (setRemoteRenderMode)

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

This method configures the remote video display mode.
The application can call this method multiple times to change the display mode.

Name	Description
uid	The user ID of the user whose video streams are received.
mode	Configures the video mode. AgoraRtcRenderMode types: ● AgoraRtc_Render_Hidden(1): If the video size is different than that of the display window, crops the borders of the video (if the video is bigger) or stretch the video ((if the video is smaller) to fit it in the window. ● AgoraRtc_Render_Fit(2): If the video size is different than that of the display window, resizes the video proportionally to fit the window. ● AgoraRtc_Render_Adaptive(3)：If both callers use the same screen orientation, that is, both use vertical screens or both use horizontal screens, the AgoraRtc_Render_Hidden mode applies; if they use different screen orientations, that is, one vertical and one horizontal, the AgoraRtc_Render_Fit mode applies.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Switch Between Front and Back Cameras (switchCamera)

```
(int)switchCamera;
```

This method switches between front and back cameras.

Name	Description
Return Value	**0：Method call succeeded.** **<0：Method call failed.**

Mute Local Video Stream (muteLocalVideoStream)

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

This method enables/disables sending local video stream to the network.

Note: When set to Yes, this method does not disable the camera and thus does not affect the retrieval of local video stream.

Name	Description
mute	**Yes: Stops sending local video stream to the network.** No: Allows sending local video stream to the network.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Mute All Remote Video Streams (muteAllRemoteVideoStreams)

(int)muteAllRemoteVideoStreams:(BOOL)muted;

This method enables/disables playing all remote callers' video streams.

Note: When set to Yes, this method stops playing video streams without affecting the video stream receiving process.

Name	Description
mute	Yes: Stops playing all received video streams. No: Allows playing all received video streams.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Mute Certain Remote Video Stream (muteRemoteVideoStream)

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

This method pauses/resumes playing a specified user's video, that is, enables/disables playing a specified user's video stream.

Note: When set to Yes, this method stops playing video streams without affecting the video stream receiving process.

Name	Description
uid	The user ID of the specified user.
mute	Yes: Stops playing a specified user's video stream. No: Allows playing a specified user's video stream.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Start Audio Recording (startAudioRecording)

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

This method starts audio recording.

The SDK allows recording during a conversation. The recording file format is .wav. Ensure that the saving directory in the application exists and is writable. This method is usually called after the joinChannel method.

Note: The recording automatically stops when the leaveChannel method is called.

Name	Description
filePath	The full file path of the recording file.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Stop Audio Recording (stopAudioRecording)

```
(int)stopAudioRecording;
```

This method stops recording on the client machine.

Note: Call this method before calling leaveChannel; otherwise the recording automatically stops when the leaveChannel method is called.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Set Log File (setLogFile)

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

This method specifies the SDK output log file.

The log file records all the log data of the SDK's operation. Ensure that the saving directory in the application exists and is writable.

Name	Description
filePath	The full file path of the log file.
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Set Log Filter (setLogFilter)

(int)setLogFilter:(NSUInteger)filter;

This method sets the SDK output log filter. You can use one filter or a combination of the filters.

Name	Description
filter	Filters • 1: INFO • 2: WARNING • 4: ERROR • 8: FATAL • 0x800: DEBUG
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Start Server Recording Service (startRecordingService)

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

This method starts the server recording function. It requires a correct Dynamic key to start the service. This method is called asynchronously，and the execution result (successful or failed) returns in the didApiCallExecute callback.

Name	Description
key	Dynamic Key
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Stop Server Recording Service (stopRecordingService)

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

This method stops the server recording function. It requires a correct Dynamic Key to start the service. This method is called asynchronously，and the execution result (successful or failed) returns in the didApiCallExecute callback.

Name	Description
key	Dynamic Key
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Refresh Server Recording Service Status (refreshRecordingServiceStatus)

```
(int)refreshRecordingServiceStatus
```

This method refreshes the server recording service status. This method is called asynchronously. The successful execution result returns in the didRefreshRecordingServiceStatus callback and unsuccessful execution result returns in the didApiCallExecute callback.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Destroy Engine Instance (destroy)

public void destroy()
This method releases all the resources being used by the Agora SDK, and may be useful for applications that perform voice or video communications from time to time according to user needs, but at other times want to maintain resources for other application functions. Once the application has called destroy() to destroy the created RtcEngine instance, no other methods in the SDK can be used and no callbacks will occur. To start communications again a new Initialize ((sharedEngineWithVendorKey) must be performed to establish a new AgoraRtcEngineKit instance according to this document.

Name	Description
Return Value	**0**：Method call succeeded. **<0**：Method call failed.

Delegate Methods（AgoraRtcEngineDelegate）

The SDK uses Delegate methods to report runtime events to the application. From version 1.1 on, some Block callbacks in the SDK are replaced with Delegate. The old Block callbacks are therefore being depreciated, but can still be used in current versions. However, we recommend replacing them with Delegate methods. The SDK calls the Block method if a callback is defined in both Block and Delegate.

Warning Occurred Callback (didOccurWarning)

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

This callback method indicates that some warning occurred during the runtime of the SDK. In most cases the application can ignore the warnings reported by the SDK because the SDK usually can fix the issue and resume running. For instance, the SDK may report an AgoraRtc_Error_OpenChannelTimeout(106) warning upon disconnection with the server, and in the meantime the SDK will attempt to reconnect.

Name	Description
warningCode	The warning code.

Error Occurred Callback (didOccurError)

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

This callback indicates that a network or media error occurred during the runtime of the SDK. In most cases reporting an error means that the SDK cannot fix the issue and resume running, and therefore requires actions from the application or simply informs the user on the issue. For instance, the SDK reports an AgoraRtc_Error_StartCall(1002) error when failed to initialize a call. In this case, the application informs the user that the call initialization failed and in the same time it also calls the leaveChannel method to exit the channel.

Name	Description
Engine	The AgoraRtcEngineKit Object.
errorCode	Error code.

Join Channel Callback (didJoinChannel)

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

Same as joinSuccessBlock in the joinChannelByKey API. This callback indicates that the user has successfully joined the specified channel.

Name	Description
channel	The channel name.
uid	The user ID. If the uid is specified in the joinChannel method, returns the specified ID; if not, returns an ID that is automatically allocated by the Agora server.
elapsed	The time elapsed in milliseconds from calling joinChannel until this event occurs.

Rejoin Channel Callback (didRejoinChannel)

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

In times when the client machine loses connection with the server because of network problems, the SDK automatically attempts to reconnect, and then triggers this callback method upon reconnection, indicating that the user has rejoined the channel with assigned channel ID and user ID.

Name	Description
channel	The channel name.
uid	The user ID.
elapsed	The time elapsed in milliseconds from calling joinChannel until this event occurs.

Audio Volume Indication Callback (reportAudioVolumeIndicationOfSpeakers)

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

Same as audioVolumeIndicationBlock.
By default the indication is disabled. If needed, use the enableAudioVolumeIndication method to configure it.

Name	Description
speakers	The speakers (array). Each speaker (): • uid: The user ID of the speaker (that is, the user who is speaking). • volume：The volume of the speaker between 0 (lowest volume) to 255 (highest volume).
totalVolume	The total volume after audio mixing between 0 (lowest volume) to 255 (highest volume).

First Local Video Frame Displayed Callback (firstLocalVideoFrameWithSize)

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

Same as firstLocalVideoFrameBlock. Indicates that the first local video frame is displayed on the video window.

Name	Description
size	The size of the video (width and height).
elapsed	The time elapsed in milliseconds from calling joinChannel until this callback is triggered.

First Remote Video Frame Received and Decoded Callback (firstRemoteVideoDecodedOfUid)

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

Same as firstRemoteVideoDecodedBlock. Indicates that the first remote video frame is received and decoded.

Name	Description
uid	The user ID of the user whose video stream is received.
size	The size of the video (width and height).
elapsed	The time elapsed in milliseconds from calling joinChannel until this callback is triggered.

First Remote Video Frame Displayed Callback (firstRemoteVideoFrameOfUid)

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

Same as firstRemoteVideoFrameBlock. Indicates that the first remote video frame is displayed on the screen.

Name	Description
uid	The user ID of the user whose video stream is received.
size	The size of the video (width and height).
elapsed	The time elapsed in milliseconds from calling joinChannel until this callback is triggered.

Other User Joined Callback (didJoinedOfUid)

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

Same as userJoinedBlock.This callback indicates that a user has successfully joined the channel. If there are other users in the channel when this user joins, the SDK also reports to the application on the existing users who are already in the channel.

Name	Description
uid	The user ID. If the uid is specified in the joinChannel method, returns the specified ID; if not, returns an ID that is automatically allocated by the Agora server.
elapsed	The time elapsed in milliseconds from calling joinChannel until this callback is triggered.

Other User offline Callback (didOfflineOfUid)

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

Same as userOfflineBlock. This callback indicates that a user has left the call or gone offline.

Note: The SDK reads the timeout data to determine if a user has left the channel (or has gone offline) or not. If no data package is received from the user in 15 seconds, the SDK takes it as the user being offline. Sometimes a weak network connection may lead to false detection, therefore we recommend using signaling for reliable offline detection.

Name	Description
uid	The user ID.
reason	Reasons for user going offline: • USER_OFFLINE_QUIT: The user has quit the call. • USER_OFFLINE_DROPPED: The SDK is timeout and dropped offline because it hasn't received data package for too long. **Note**: Sometimes when the other user quits the call but the message is not passed to the SDK due to unreliable channel, the SDK may mistake it as the other user is timeout and dropped offline.

Other User Muted Audio Callback (didAudioMuted)

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

Same as userMuteAudioBlock。This callback indicates that some other user has muted/unmuted his/her audio streams.
Note: This callback method is valid only when the channel profile is set to Free-talk mode.

Name	Description
uid	The user ID.
muted	Yes：The user has muted his/her audio. No：The user has unmuted his/her audio.

Other User Paused/Resumed Video Callback (didVideoMuted)

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

Same as userMuteVideoBlock. This callback indicates that some other user has paused/resumed his/her video streams.

Name	Description
uid	The user ID.
muted	Yes：The user has paused his/her video. No：The user has resumed his/her video.

Other User Enabled/Disabled Video Callback(didVideoEnabled)

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

This callback indicates that other users enabled/disabled the video function. Once the video function is disabled, users can only perform audio call. They cannot see any video from both themselves and other users.

Name	Description
uid	User ID
enabled	Yes: The user has enabled the video function. No: The user has disabled the video function.

Local Video Statistics Callback (localVideoStats)

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

Same as localVideoStatBlock.The SDK updates the application on local video streams uploading statistics once every 2 seconds.

Name	Description
sentBytes	Total bytes sent since last count.
sentFrames	Total frames sent since last count.

Remote Video Statistics Callback (remoteVideoStatOfUid)

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

Same as remoteVideoStatBlock.The SDK updates the application on the statistics about receiving remote video streams once every 2 seconds.

Name	Description
uid	The user ID of the user whose video streams are received.
delay	The time delay (in milliseconds).
width	The width (in pixel) of the video stream.
height	The height (in pixel) of the video stream.
receivedBitrate	The data receive bitrate (in kbps) since last count.
receivedFrameRate	The data receive frame rate (in fps) since last count.

Camera Ready Callback (rtcEngineCameraDidReady)

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

Same as cameraReadyBlock. This callback indicates that the camera is opened and ready to capture video.

Video Stopped Callback (rtcEngineVideoDidStop)

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

This callback indicates that the video is stopped. The application can use this callback to change the configuration of the view (for example, displaying other pictures on the view) after the video stops.

Connection Interrupted Callback (rtcEngineConnectionDidInterrupted)

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

This callback indicates that connection is lost between the SDK and server. Once the connection is lost, the SDK tries to reconnect it repeatedly until the application calls `leaveChannel`.

Connection Lost Callback (rtcEngineConnectionDidLost)

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

When the connection between the SDK and server is lost, the rtcEngineConnectionDidInterrupted callback is triggered and the SDK reconnects automatically. If the reconnection fails within a certain period（10s by default）, the callback *rtcEngineConnectionDidLost* will be triggered。The SDK continues to reconnect until the application calls leaveChannel

Rtc Engine Statistics Callback (reportRtcStats)

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

Same as rtcStatsBlock. The SDK updates the application on the statistics of the Rtc Engine runtime status once every 2 seconds.

Name	Description
stat	See table below.

AgoraRtcStats	Description
duration	The call duration in seconds, represented by an aggregate value.
txBytes	The total number of bytes transmitted, represented by an aggregate value.
rxBytes	The total number of bytes received, represented by an aggregate value.

Audio Quality Callback (audioQualityOfUid)

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

Same as audioQualityBlock. During a call, this callback is triggered once every 2 seconds to report on the audio quality of the current call.

Name	Description
audioQualityBlock	
uid	The user ID. Each callback reports on the audio quality of one user only (excluding the user himself). Every different user's statistics is reported in a separate callback.
quality	The rating of the audio quality: 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	The time delay in milliseconds.
lost	The packet loss rate (in percentage).

Network Quality Callback (networkQuality)

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

Same as networkQualityBlock. This callback is triggered once every 2 seconds during a call to report on the network quality.

Name	Description
quality	The rating of the network quality: 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

Recording Service Status Refreshed Callback (didRefreshRecordingServiceStatus)

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

This callback returns the status code after executing the refreshRecordingServiceStatus method successfully.

Name	Description
Status Value	0：Recording is stopped. 1：Recording is ongoing.

Recording Started/Ended/Status Returned (didApiCallExecute)

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

This callback returns the status code after a successful or unsuccessful execution of the startRecordingService or stopRecordingService method. It also returns the status code after an unsuccessful execution of the `refreshRecordingServiceStatus` method.

Method	API String Parameter	Status Code
StartRecordingService	rtc.api.start_recording_service	0：AgoraRtc_Error_NoError 1：AgoraRtc_Error_Failed 2: AgoraRtc_Error_InvalidArgument 3: ERR_NOT_READY 7：AgoraRtc_Error_NotInitialized 10: AgoraRtc_Error_Timedout
stopRecordingService	rtc.api.stop_recording_service	
refreshRecordingServiceStatus	rtc.api.query_recording_service_status	The code can be one of the above 1,2,3 ,7 or 10. The description is the same as above.

Callback Methods

The SDK supports Block to report runtime events to the application. However, see the previous section on delegate methods. From version 1.1 on, some Block callbacks in the SDK are replaced with Delegate. The old Block callbacks are therefore being depreciated, but can still be used in current versions. However, we recommend replacing them with Delegate methods. The SDK calls the Block method if a callback is defined in both Block and Delegate.

The following section describes each of the callback methods in the SDK.

Other User Joined Callback (userJoinedBlock)

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

This callback indicates that a user has successfully joined the channel. If there are other users in the channel when this user joins, the SDK also reports to the application on the existing users who are already in the channel.

Name	Description
uid	The user ID. If the uid is specified in the joinChannel method, returns the specified ID; if not, returns an ID that is automatically allocated by the Agora server.
elapsed	The time elapsed in milliseconds from calling joinChannel until this callback is triggered.

Other User offline Callback (userOfflineBlock)

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

This callback indicates that a user has left the call or gone offline.
Note: The SDK reads the timeout data to determine if a user has left the channel (or went offline) or not. If no data package is received from the user in 15 seconds, the SDK takes it as the user being offline. Sometimes a weak network connection may lead to false detection, therefore we recommend using signaling for reliable offline detection.

Name	Description
uid	The user ID.

Rejoin Channel Callback (rejoinChannelSuccessBlock)

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

In times when the local machine loses connection with the server because of network problems, the SDK automatically attempts to reconnect, and then triggers this callback method upon reconnection. The callback indicates that the user has rejoined the channel with an allocated channel ID and a user ID.

Name	Description
channel	The channel name.
uid	The user ID.
elapsed	Time delay (in milliseconds) in rejoining the channel.

User Left Channel Callback (leaveChannelBlock)

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

This callback indicates that a user has left the channel, and it also provides call session statistics including the duration of the call and tx/rx bytes.

Name	Description
stat	See table below.

AgoraRtcStats	Description
duration	The call duration in seconds, represented by an aggregate value.
txBytes	The total number of bytes transmitted, represented by an aggregate value.
rxBytes	The total number of bytes received, represented by an aggregate value.

Rtc Engine Statistics Callback (rtcStatsBlock)

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

This callback is triggered once every 2 seconds to update the application on Rtc Engine runtime statistics.

Name	Description
stat	See table above.

Audio Volume Indication Callback (audioVolumeIndicationBlock)

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

This callback indicates who is speaking and the speaker's volume. By default it is disabled.
If needed, use the enableAudioVolumeIndication method to configure it.

Name	Description
speakers	The speakers (array). Each speaker (): ● uid: The user ID of the speaker (that is, the user who is speaking). ● volume： The volume of the speaker between 0 (lowest volume) to 255 (highest volume).
totalVolume	The total volume after audio mixing between 0 (lowest volume) to 255 (highest volume).

First Local Video Frame Displayed Callback (firstLocalVideoFrameBlock)

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

This callback indicates that the first local video frame is displayed on the screen.

Name	Description
width	The width (in pixel) of the video stream.
height	The height (in pixel) of the video stream.
elapsed	The time elapsed in milliseconds from the user joining the channel until this callback is triggered.

First Remote Video Frame Displayed Callback (firstRemoteVideoFrameBlock)

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

This callback indicates that the first remote video frame is displayed on the screen.

Name	Description
uid	The user ID of the user whose video streams are received.
width	The width (in pixel) of the video stream.
height	The height (in pixel) of the video stream.
elapsed	The time elapsed in milliseconds from the user joining the channel until this callback is triggered.

First Remote Video Frame Received and Decoded Callback (firstRemoteVideoDecodedBlock)

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

This callback indicates that the first remote video frame is received and decoded.

Name	Description
uid	The user ID of the user whose video streams are received.
width	The width (in pixel) of the video stream.
height	The height (in pixel) of the video stream.
elapsed	The time elapsed in milliseconds from the user joining the channel until this callback is triggered.

Other User Muted Audio Callback (userMuteAudioBlock)

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

This callback indicates that a user has muted/unmuted his/her audio stream.

Name	Description
uid	The user ID.
muted	Yes：Muted audio. No: Unmuted audio.

Other User Paused/Resumed Video Callback (userMuteVideoBlock)

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

This callback indicates that a user has paused/resumed his/her video stream.

Name	Description
uid	The user ID
muted	Yes：The user has paused his/her video. No: The user has resumed his/her video.

Local Video Statistics Callback (localVideoStatBlock)

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

The SDK updates the application about the statistics on uploading local video streams once every 2 seconds.

Name	Description
sentBytes	The total bytes sent since last count.
sentFrames	The total frames sent since last count.

Remote Video Statistics Callback (remoteVideoStatBlock)

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

The SDK updates the application about the statistics on receiving remote video streams once every 2 seconds.

Name	Description
uid	The user ID that specifies which user's video streams are received.
frameCount	The total frames received since last count.
delay	The time delay in milliseconds.
receivedBytes	The total bytes received since last count.

Camera Ready Callback (cameraReadyBlock)

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

This callback indicates that the camera is opened and ready to capture video.

Audio Quality Callback (audioQualityBlock)

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

During a conversation, this callback is triggered once every 2 seconds to report on the audio quality of the current call.

Name	Description
audioQualityBlock	
uid	The user ID. Each callback reports on the audio quality of one user only (excluding the user himself). Every different user's statistics is reported in a seperate callback.
quality	The rating of the audio quality: 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	The time delay in milliseconds.
lost	The packet loss rate (in percentage).

Network Quality Callback (networkQualityBlock)

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

This callback s triggered once every 2 seconds during a call to report on the network quality.

Name	Description
quality	The rating of network quality: 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

Connection Lost Callback (connectionLostBlock)

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

This callback indicates that the SDK has lost network connection with the server.