Agora Windows SDK Reference

support@agora.io

Contents

Required Libraries

- Agora Audio SDK requires Visual C++ 2008 x86 runtime libraries.
- Add the AgoraAudioSDK/include directory to the INCLUDE directories of your project.
- Add the 'AgoraAudioSDK/lib' directory to the LIB directories of your project and make sure mediasdk.lib is linked with your project.
- Copy .dlls under AgoraAudioSDK/dll to the directory where your executable file is located.

AgoraAudio Methods

Create Agora Audio Object

AgoraAudio = createAgoraAudioInstance(IAgoraAudioEventHandler* pHandler);

This statement initializes the AgoraAudio class.

Name	Description
pHandler	IAgoraAudioEventHandler (See below)
Return Value	IAgoraAudio object

Release Agora Audio Object
This method destroys the AgoraAudio object.

void release()

Join Channel

This method lets users join a channel. Think of it as joining a chat room, except that it is a multi-party phone call. This method is asynchronous, so it can be called on the main UI thread.

void joinChannel(const char* vendorKey, const char* channelName, const char* info, unsigned int uid)

Name	Description
vendorKey	Account credentials issued by Agora Voice to app developer, i.e., a user license.
channelName	Channel name. Arbitrary descriptor like “game1” or “call2”.

info	Optional. Whatever the additional information the programmer wants to add.
uid	Optional. User id. If you do not set this parameter, the object instance will automically generate one.

Leave Channel

Leave channel, meaning hang up or exit call. Returns 0 if the client successfully left the channel.

int leave()

Set Parameters

void setParameters(const char* parameters)

The following are available parameters used for the Agora Audio engine. The parameters must be in JSON format when specifying new parameters to set. Instead of being called directly by the app, it is usually called by the helper class AgoraAudioParameters.

Name	Description
parameters	Parameters in JSON format: mute mutePeers speakerOn speakerVolume micVolume enableVolumeReport volumeSmoothFactor logFilter

Get Parameters

int getParameters(const char* parameters, char* buffer, size_t* length)

Retrieve the current parameters settings.

Name	Description
parameters	Indicate which parameters to retrieve
buffer	String containing values

length	Length of buffer

Get Call ID

const char* getCallId()

Retrieve current call ID.

Rate For The Call

int rate(const char* callId, int rating)

This method lets the user rate the call.

Name	Description
callId	Call ID got from getCallId method
Rating	Rating for the call, range from 1(min) to 10(max)
Return Value	0: successful method call -1: method call failed

Complain About The Call

int complain(const char* callId)

This method lets the user complain about the call.

Name	Description
callId	Call ID got from getCallId method
Return Value	0: successful method call -1: method call failed

Start Echo Test

int StartEchoTest(const char* key)

This method lets the user launch an echo test, paired with StopEchoTest. In an echo test, the user says something and then it's played back so that call connectivity and quality can be tested. (NOTE: echo test should be stopped before starting a call)

Name	Description
key	Account credentials issued by Agora Voice to app developer, i.e., a user

Return Value	0: successful method call -1: method call failed

Stop Echo Test

int StopEchoTest()

This method stops an echo test.

Name	Description
Return Value	0: successful method call -1: method call failed

Enable Network Test

int enableNetworkTest(const char* key)

This method launches a test to monitor the real-time user network quality. The network test will stop when the app is put into a background state. (NOTE: Network testing is off by default)

Name	Description
key	Account credentials issued by Agora Voice to app developer, i.e., a user
Return Value	0: successful method call -1: method call failed

Disable Network Test

int disableNetworkTest()

This method disables the network test.

Name	Description
Return Value	0: successful method call -1: method call failed

Make Call Quality Report URL

int makeQualityReportUrl(const char* vendorKey, const char* channel, uid_t listenerUid, uid_t speakerUid, int format, char* buffer, size_t* length)

This method lets the user retrieve the call quality report URL.

Name	Description
vendorKey	Account credentials issued by Agora Voice to app developer, i.e., a user
channel	The channel number specified in joinChannel method.
listenerUid	The user ID - This could be the uid specified in joinChannel method. If it is not set, the SDK object instance will allocate one and the user can retrieve this ID from onJoinSuccess callback (see below).
speakerUid	The speaker ID - The user can retrieve it from the onUserJoined callback (see below). If speakerUid is not set, the method call will request a report all speakers in the group call.
format	Report format type 0: JSON 1: HTML
buffer	User specified input buffer to store the report URL
length	Input buffer length
Return Value	0: successful method call -1: method call failed

AgoraAudioParameters Methods

Mute

void mute(bool mute)

Turns off the microphone.

Name	Description
mute	True - turns microphone off False - turns microphone on

Mute All Speakers

void mutePeers(bool mute);

Turns off both the speaker and earpiece.

Name	Description
mute	True - turns off all audio output False - turns on all audio output

Mute Specific User

void mutePeer(bool mute, unsigned int uid);

Turn off audio for a specific caller.

Name	Description
mute	True - muted, False - unmuted (i.e., caller's microphone turned back on)
Uid	ID of user to mute

Set Speaker Volume

void setSpeakerVolume(int volume);

Name	Description
volume	Set speaker volume from 0 (min) to max (255)

Set Microphone Volume

void setMicrophoneVolume(int volume);

Name	Description
volume	Set microphone volume from 0 (min) to max (255)

IAgoraAudioEventHander Interface Methods

The following callback methods, within IaudioEventHandler, are executed upon events such as: joining a call, error reports, call quality reports, etc.

onLoadAudioEngineSuccess

virtual void onLoadAudioEngineSuccess() = 0;

You should implement this method to indicate what happens when the Agora audio engine is loaded correctly. This means the app was able to connect to an available audio server. From this point the audio engine is working, meaning it

is in communication mode. This is the ideal location in your code to create a time marker, allowing you to calculate and display call duration.

onJoinSuccess

virtual void onJoinSuccess(const char* channel, uid_t uid, int elapsed) = 0;

Indicates that the server has authenticated the client. The channel id is based on the channel name parameter specified by the join() API method. If the user id was not specified with the call to join(), the server will automatically allocate one.

Name	Description
channel	Channel id
uid	User id
elapsed	Delay in ms

onRejoinSuccess

virtual void onRejoinSuccess(const char* channel, uid_t uid, int elapsed) = 0;

Indicates that the client has rejoined the server. Like onJoinSuccess(), the channel id and user id are passed back.

Name	Description
channel	Channel id
uid	User id
elapsed	Delay in ms

onError

virtual void onError(int rescode, const char* msg) = 0;

Indicates that an error occurred in the audio engine or from the network connection.

Name	Description
rescode	EVENT_LOAD_AUDIO_ENGINE_ERROR = 1001: failed to initialize audio engine EVENT_START_CALL_ERROR = 1003: failed to start audio engine. Typically this is caused because the audio device is in use by another app EVENT_JOIN_GET_AUDIO_ADDR_TIMEOUT = 11002:

	voice server list timeout EVENT_JOIN_GET_AUDIO_ADDR_FAILED = 11003: error code received when requesting voice server list EVENT_JOIN_GET_AUDIO_ADDR_ZERO_ADDR = 11004: The Voice Center Server (acts as a gateway, like DNS) response that a voice server is not available EVENT_JOIN_CONNECT_MEDIA_TIMEOUT = 12002: connection to voice server timeout EVENT_JOIN_LOGIN_MEDIA_TIMEOUT_ALL = 13003: login to voice server timeout EVENT_JOIN_LOGIN_MEDIA_FAILED = 13004: Failed to login to voice sever, server ACKed with error code EVENT_JOIN_LOGIN_REGET_AUDIO_ADDR = 13005: the voice central server (acts as a gateway, like DNS) tried all available voice servers but none are able to accept this call
msg	Error message that you want to display to the end-user or use for other purposes

onAudioQuality

virtual void onAudioQuality(unsigned int uid, int quality, unsigned short delay, unsigned short jitter, unsigned short lost, unsigned short lost2) = 0;

Reports audio quality. This callback function will be triggered every 2seconds.

Name	Description
uid	User id, i.e. the caller.
quality	Voice quality rating MEDIA_QUALITY_EXCELLENT = 1 MEDIA_QUALITY_GOOD = 2 MEDIA_QUALITY_POOR = 3 MEDIA_QUALITY_BAD = 4

	MEDIA_QUALITY_VBAD = 5 MEDIA_QUALITY_DOWN = 6
delay	Voice delay in ms
jitter	Jitter is the delay of inbound packets, due to network congestion or queuing issues
lost	Packet loss ratio
lost2	Number of times that 2 consecutive packets were lost

onNetworkQuality

virtual void onNetworkQuality(int quality) = 0;

Reports network quality. This callback function will be triggered every 2 seconds.

Name	Description
quality	Network quality rating MEDIA_QUALITY_EXCELLENT = 1 MEDIA_QUALITY_GOOD = 2 MEDIA_QUALITY_POOR = 3 MEDIA_QUALITY_BAD = 4 MEDIA_QUALITY_VBAD = 5 MEDIA_QUALITY_DOWN = 6

onLeaveChannel

virtual void onLeaveChannel(const SessionStat& stat) = 0;

Indicates the user left the channel. This callback provides session statistics information including call duration and tx/rx bytes.

Name	Description

stat	struct SessionStat {
	unsigned int duration;
	unsigned int txBytes;
	unsigned int rxBytes;
	};

onUserJoined

virtual void onUserJoined(uid_t uid, int elapsed) = 0;

Indicates that a particular user joined. This callback provides the user's id and join delay in ms.

Name	Description
uid	User id
elapsed	Delay in ms

onUserOffline

virtual void onUserOffline(uid_t uid) = 0;

Indicates that a particular user has left the call.

Name	Description
uid	User id

onUpdateSessionStats

virtual void onUpdateSessionStats(const SessionStat& stat) = 0;

This callback provides updated session statistics information. This callback function will be triggered every 2 seconds.

Name	Description

stat	struct SessionStat {
	unsigned int duration;
	unsigned int txBytes; //
	transmission
	unsigned int rxBytes; // receipt
	};

onAudioEngineEvent

virtual void onAudioEngineEvent(int evt) = 0;

This callback provides audio engine event messages.

Name	Description
evt	enum AUDIO_ENGINE_EVENT_CODE { AUDIO_ENGINE_RECORDING_ERROR = 0, // recording cannot proceed

	AUDIO_ENGINE_PLAYOUT_ERROR = 1, // player cannot proceed
	
	AUDIO_ENGINE_RECORDING_WARNING = 2, // other recorder related events
	
	AUDIO_ENGINE_PLAYOUT_WARNING = 3 // other player related events
	};

onAudioDeviceStateChanged

virtual void onAudioDeviceStateChanged(const char* deviceId, int deviceType, int deviceState) = 0;

Indicates that the system's audio device state has changed, such as earphones unplugged from the device

Name	Description
deviceId	device id identifying an audio device

deviceType	enum AUDIO_DEVICE_TYPE {
	UNKNOWN_AUDIO_DEVICE = -1,
	PLAYOUT_DEVICE = 0,
	RECORDING_DEVICE = 1
	};

deviceState	enum AUDIO_DEVICE_STATE_TYPE {
	AUDIO_DEVICE_STATE_ACTIVE = 1,
	
	AUDIO_DEVICE_STATE_DISABLED = 2,
	
	AUDIO_DEVICE_STATE_NOT_PRESENT = 4,
	
	AUDIO_DEVICE_STATE_UNPLUGGED = 8
	};