



Agora Windows How-to Guide

support@agora.io

Contents

| | |
|--------------------------------------|---|
| Requirements..... | 3 |
| Demonstration..... | 3 |
| Sample Program..... | 7 |
| Create AgoraAudio object..... | 7 |
| Join Call | 7 |
| Leave Call..... | 7 |
| Release AgoraAudio object..... | 7 |
| AgoraAudio Interface | 7 |
| AudioEventParameters Interface | 8 |
| IAudioEventHandler Interface | 8 |

This how-to guide contains a quick tutorial on how to get started with using Agora's SDK; for your convenience, the SDK includes a sample executable demo.

Requirements

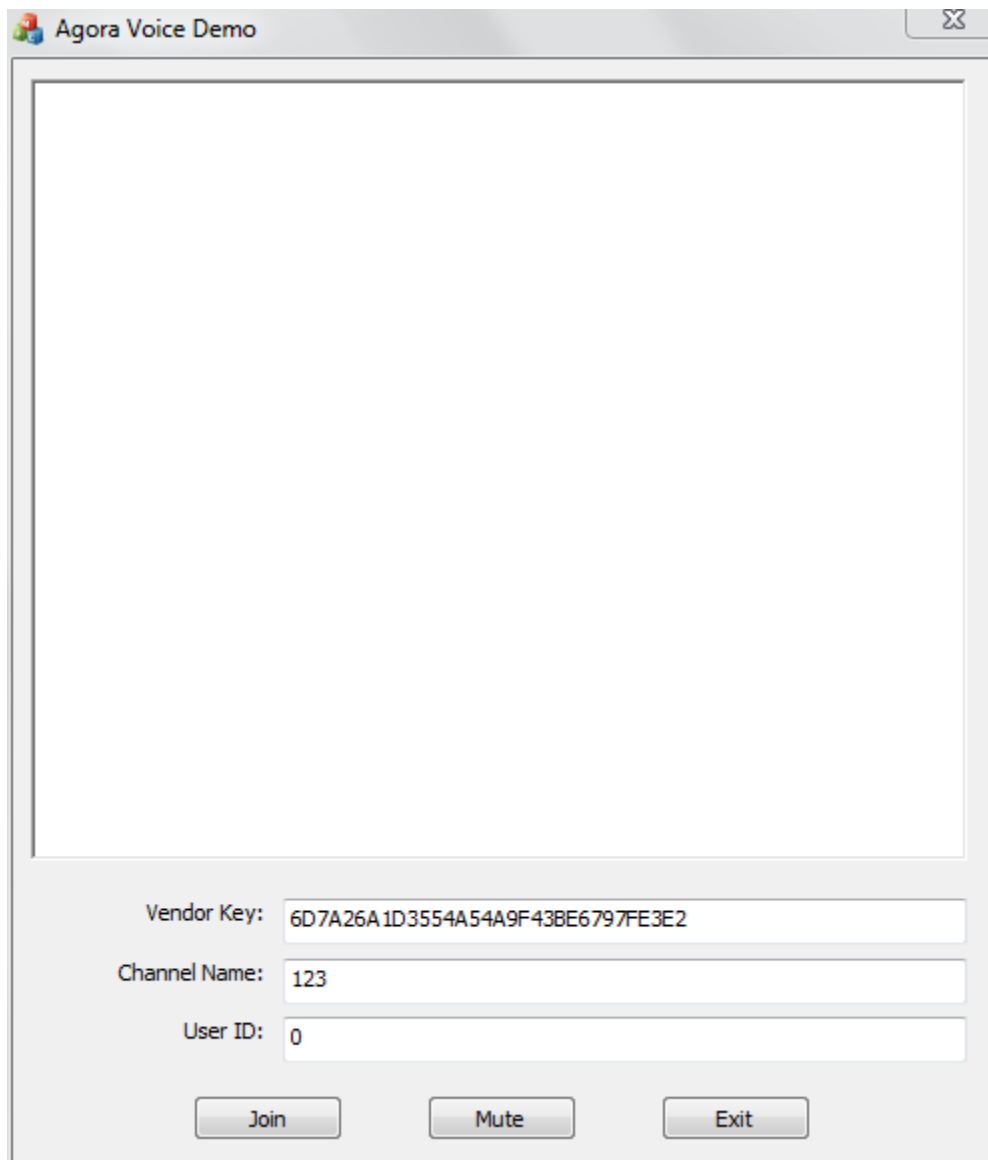
- Microsoft Visual C++ 2008 or greater
- Windows Mobile Professional Developers toolkit
- Windows mobile phone OR Windows mobile emulator. (NOTE: Having two phones is highly recommended to allow calling from one test phone to another.
- Add the AgoraAudioSDK/include directory to the INCLUDE directories 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 the .dlls under AgoraAudioSDK/dll to the directory where your executable file is located.

Demonstration

There are four basic functions to the Agora Voice SDK:

1. Initialization
2. Join Channel
3. Leave Channel
4. Release

We will go into further detail in the following sections below with code samples, but first, we recommend that you run the demo to see the events lifecycle and their respective results.

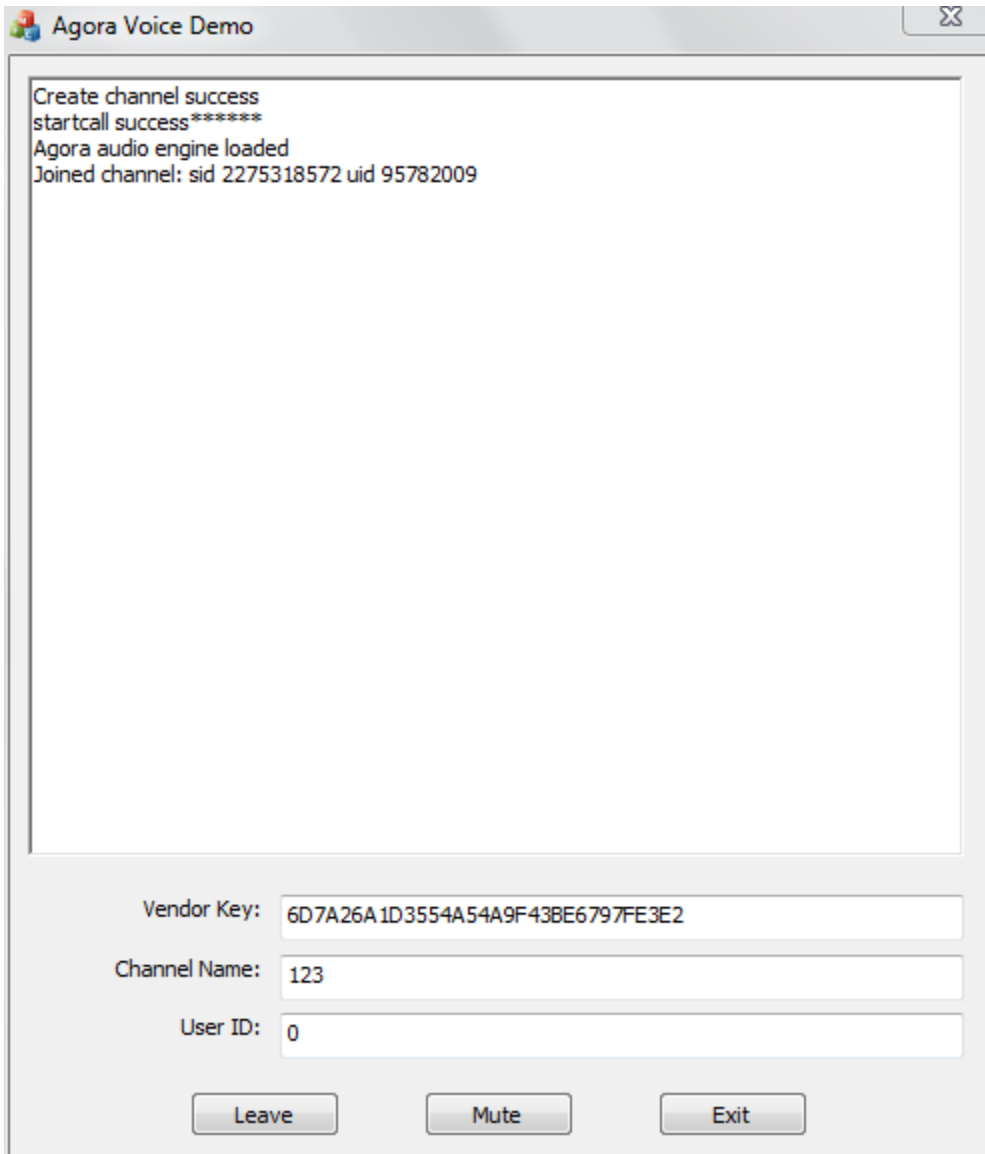


Enter the three required fields listed below, then press “Join” to join the call on the desired channel. You can do this even if no one else is in the channel, meaning you are the first person to join the call.

Vendor Key—this is the license key supplied to the Agora Voice customer to make calls over the Agora Voice cloud.

Channel Name—this is the name of the channel you will join. In the example below we use “1,” but it can be something arbitrary, like “conference call” or “game XYZ”.

User ID—if you leave this blank, the AgoraVoice SDK object instance will automatically create an ID for you. This uniquely identifies a member of the call. For this demo, you may leave it blank.



The screenshot shows a window titled "Agora Voice Demo". Inside, there is a text area with the following log messages:

```
Create channel success
startcall success*****
Agora audio engine loaded
Joined channel: sid 2275318572 uid 95782009
```

Below the log, there are three input fields:

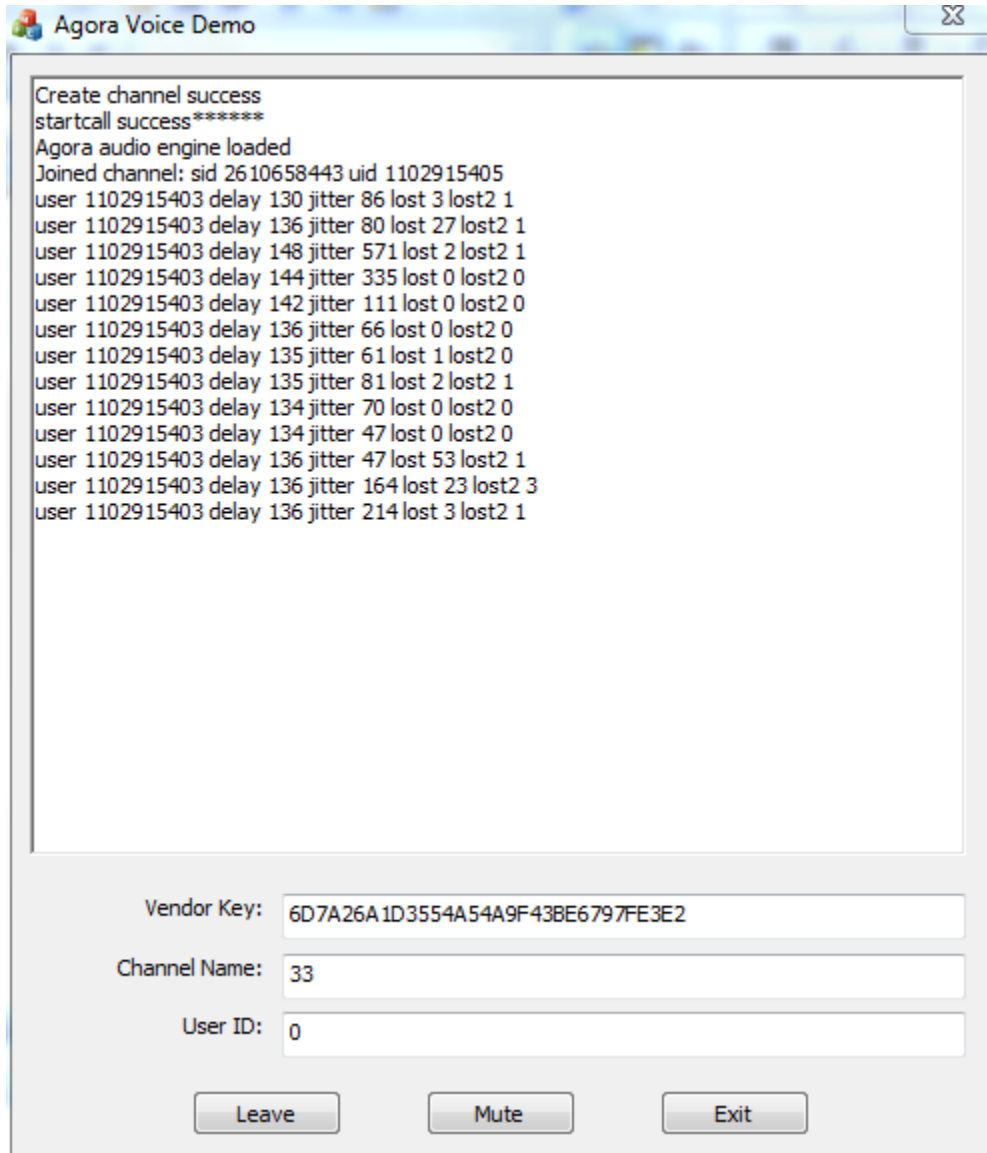
- Vendor Key: 6D7A26A1D3554A54A9F438E6797FE3E2
- Channel Name: 123
- User ID: 0

At the bottom of the window, there are three buttons: "Leave", "Mute", and "Exit".

Channel joined successfully.

sid—this is integer representation of the channel number that you entered, which in this case is “123.”

uid—User id. Since you entered “0”, the uid was automatically generated.



Now that you have joined the channel, the screen will display a log of call quality and latency data. When a second person joins the call, you will see the same log data from that caller as well.

delay—voice latency/delay in ms.

Jitter—VoIP jitter occurs when the data packets sent from a client are sent and received with significant timing differences.

lost— loss ratio in percentage, from 0 to 99.

Sample Program

Shown below are four basic operations and code samples for: initialize, join, leave, and mute.

Create AgoraAudio object

```
IAgoraAudio* pAgoraAudio =  
    createAgoraAudioInstance(IAgoraAudioEventHandler);
```

See information below on the IAgoraAudioEventHandler.

Join Call

```
const char* key = "key granted by Agora";  
const char* channel = "channel to join";  
const char* extraInfo = "extra info you pass to SDK";  
unsigned int uid = (put 0 to have Agora create a user id);  
pAgoraAudio->joinChannel(key, channel, extraInfo, uid);
```

Leave Call

```
pAgoraAudio->leave();
```

Release AgoraAudio object

```
pAgoraAudio->release();
```

AgoraAudio Interface

For more AgoraAudio SDK API methods listed below, please refer to the SDK Reference API document for more details.

```
pAgoraAudio ->setParameters(const char* parameters);  
  
pAgoraAudio ->getParameters(const char* parameters, char* buffer,  
size_t* length);  
  
pAgoraAudio-> getCallId();  
  
pAgoraAudio-> rate(const char* callId, int rating);  
  
pAgoraAudio-> complain(const char* callId);
```

```

pAgoraAudio-> StartEchoTest(const char* key);

pAgoraAudio-> StopEchoTest();

pAgoraAudio-> enableNetworkTest(const char* key);

pAgoraAudio-> disableNetworkTest();

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

```

AudioEventParameters Interface

Create [AudioEventParameters](#) object:

```
AudioEventParameters pEventParam(pAgoraAudio);
```

The [AudioEventParameters](#) has an interface allowing you to set audio engine parameters (e.g. muting and setting device volume). See the SDK Reference documents for more details.

```

pEventParam.mute(bool mute);

pEventParam.mutePeers(bool mute);

pEventParam.mutePeer(bool mute, unsigned int uid);

pEventParam.setSpeakerVolume(int volume);

pEventParam.setMicrophoneVolume(int volume);

```

IAgoraAudioEventHandler Interface

The [IAgoraAudioEventHandler](#) contains virtual functions that you can implement to receive events and event data. The “onJoinSuccess” callback indicates that you are successfully logged into server. For example, the “onAudioQuality” and “onNetworkQuality” methods give you information on call/network quality. Please see the SDK Reference documents for more detailed information on these functions and parameters.

```

virtual void onLoadAudioEngineSuccess() {}

virtual void onJoinSuccess(const char* channel, uid_t uid, int elapsed) {}

virtual void onRejoinSuccess(const char* channel, uid_t uid, int elapsed) {}

```



```
virtual void onError(int rescode, const char* msg) {}  
  
virtual void onAudioQuality(unsigned int uid, int quality, unsigned short delay,  
unsigned short jitter, unsigned short lost, unsigned short lost2) {}  
  
virtual void onNetworkQuality(int quality) {}  
  
virtual void onLeaveChannel(const SessionStat& stat) {}  
  
virtual void onUserJoined(uid_t uid, int elapsed) {}  
  
virtual void onUserOffline(uid_t uid) {}  
  
virtual void onUpdateSessionStats(const SessionStat& stat) {}  
  
virtual void onAudioEngineEvent(int evt) {}  
  
virtual void onAudioDeviceStateChanged(const char* deviceId, int deviceType,  
int deviceState) {}
```