



Agora Whiteboard SDK Reference Manual v1.0

support@agora.io

Contents





Introduction	3
Agora CaaS	3
Agora Whiteboard SDK	3
Requirements.....	4
<i>Compatibility</i>	4
<i>Supported Web Browsers</i>	4
<i>Supported File Formats</i>	4
<i>Required Documents</i>	4
Known Issue and Limitation.....	5
Getting Started	6
Where to Get the SDK.....	6
About Keys	6
<i>Obtaining and Using a</i>	7
<i>Vendor Key</i>	7
<i>Obtaining and Using a Dynamic Key</i>	8
Realizing Whiteboard Function – Web Application.....	12
<i>Deploying Your Web Application</i>	12
<i>Running the Demo Application</i>	12
Realizing the Whiteboard Function – Non-Web Application	13
Agora Whiteboard SDK - API Reference	14
join(params)	14
leave(url)	15

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 in networks such as 3G/4G/Wi-Fi that perform erratically and Internet bottlenecks worldwide.



Agora CaaS includes the following SDKs and the applications using the Native SDK link it directly into the application when they are built:

-  **Agora Native SDK for iOS and Agora Native SDK for Android**
Mobile-optimized for smartphones, allowing access to the Agora Global Network along with device-specific mobile optimizations.
-  **Agora Native SDK for Windows and Agora Native SDK for Mac**
-  **Agora Native SDK for Web**
The SDK provides web browsers with open access to the Agora Global Network.
-  **Agora Whiteboard SDK**
The Agora Whiteboard SDK is an addition to the Agora CaaS capabilities. The SDK provides a simple collaboration platform on the whiteboard where users from different locations can draw, annotate, and share PDF documents to visualize and simplify the communication. The SDK provides open access to the Agora Global Network from any device that supports a standard HTML5-compliant web browser, without requiring any downloads or plugins.
See [Getting Started](#) for details about how to use the Agora Whiteboard SDK.



Agora Whiteboard SDK

The Agora Whiteboard SDK is a JavaScript library loaded by an HTML web page. It also provides a set of simple high-level JavaScript APIs for establishing whiteboard communications with users across the Agora Global Network.

The Agora Whiteboard SDK allows your JavaScript code to:

-  **Establish Session** – Join and leave the shared Agora sessions (identified by unique channel names), where there may be many global users on a conference together.
-  **Set user roles in the shared sessions** – host or attendee. Host can draw and annotate on the whiteboard, while attendees have view-only permission.

The application developers can access the Agora Whiteboard service in two ways:

-  Web Application developers can use the JavaScript Class (Agora.Whiteboard) to join or leave a session. See [Realizing Whiteboard Function – Web Application](#) and [Agora Whiteboard SDK - API Reference](#) for details.
-  Non-Web Application Developers (on any of the following platforms: iOS, Windows, Mac or Android) can use WebView to access the Agora Whiteboard URL service. See [Realizing the Whiteboard Function – Non-Web Application](#) for details.

Requirements

Compatibility

Agora Whiteboard SDK is compatible with all platform SDK versions.

Supported Web Browsers

The Agora Whiteboard SDK requires a HTML5-compliant web browser and has been tested and verified on all the following browser types and versions, which are available across a wide range of platforms.

	IE v10-11	Chrome v49-51	Safari V9.1	Firefox v46-47	Opera v38
HTTP/ HTTPS	Y	Y	Y	Y	Y

Note

All the browser versions listed have been tested and verified. Earlier versions of these browsers that support HTML5 may work, but have not been verified by Agora.io.

Supported File Formats

You can upload PDFs, office files and pictures in any of the following formats to the whiteboard:

File	Format
PDF	<ul style="list-style-type: none">• .pdf
Excel	<ul style="list-style-type: none">• .xls• .xlsx
Word	<ul style="list-style-type: none">• .doc• .docx
PowerPoint	<ul style="list-style-type: none">• .ppt• .pptx
Picture	<ul style="list-style-type: none">• .png• .jpg• .jpeg

Note

When uploading pictures, the format JPEG2000 is not supported.

Required Documents

The Whiteboard SDK only provides the whiteboard functions for users to draw, upload files and annotate. If you want to realize the ordinary video or audio call functions, refer to



Each SDK reference manual

All platform SDK and related reference manuals can be downloaded at:

agora.io/developer

Known Issue and Limitation

- Each conference can have no more than one whiteboard instance.
- The users can use the whiteboard function on the Agora supported platforms including Web, Mac and Windows. On iOS and Android, the users have view-only permission.
- If your application is on Mac, the file upload function is not supported.
This issue is caused by OS X unable to open the panel in WKWebView, and the file upload window cannot popup.
Wait for OS X to update.

Getting Started

Where to Get the SDK

The Agora Native SDK for Android is available from agora.io/developer, or contact sales@agora.io.

Component	Description
./doc	Agora Whiteboard SDK Documentation(Both English and Chinese): Agora_Whiteboard_SDK_Reference_Manual_v1_0_EN.pdf Agora_Whiteboard_SDK_Reference_Manual_v1_0_CHS.pdf
./client	Agora Whiteboard JavaScript library and web demo application
./server	Web server-side sample code for dynamic key generation

About Keys

This section describes the concepts and use of vendor, sign, and dynamic keys.

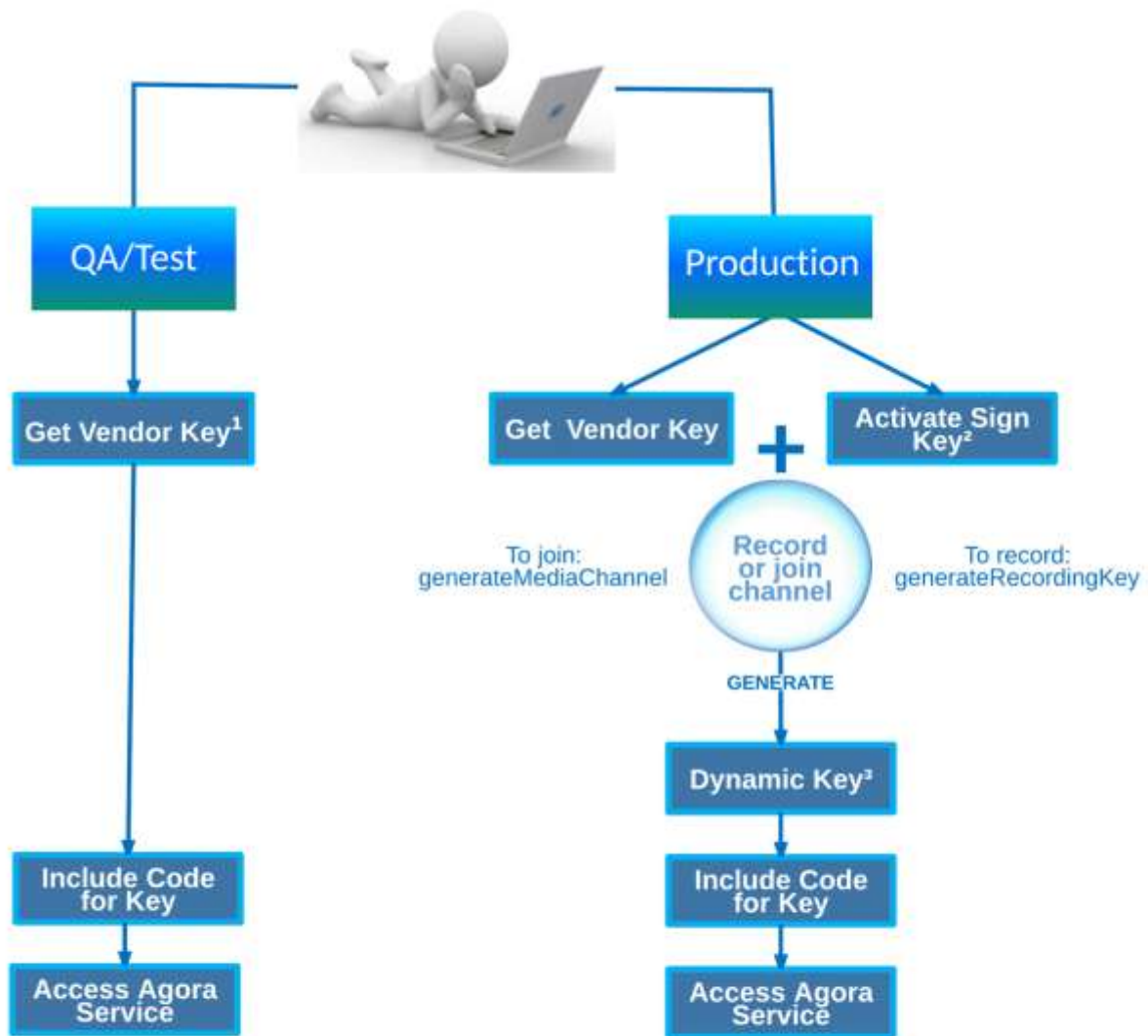
Each Agora account can create multiple projects, and each project has a unique vendor key. Be sure to obtain a **Vendor Key**, required when you use APIs to access the Agora Global Network. In the Agora network, you are isolated from others by vendor key. There is no overlap even when channel have the same name.

But vendor key is a static key, and if someone else illicitly obtained your static vendor key and then they can use it on their own Agora SDK client applications. If they find out the channel names of your organization, they can even interfere with your communication sessions.

A dynamic Key is a more secure user authentication schema for the Agora SDK. Whenever a user tries to enter a channel to access the Agora service, the back-end services use the vendor key and a sign Key to generate a new dynamic key based on the HMAC encryption algorithm. The dynamic key is then passed to the client application. The client application calls the `Agora.Whiteboard.join`(for web application) or use `WebView` to access the Agora Whiteboard URL(for non-web application) and passes the encoded dynamic key to the Agora server for the user authentication.

Note

Agora Whiteboard SDK does not support the recording function currently.



¹ Vendor key for test/lab or non-recording, low security purposes

² Cannot be used alone

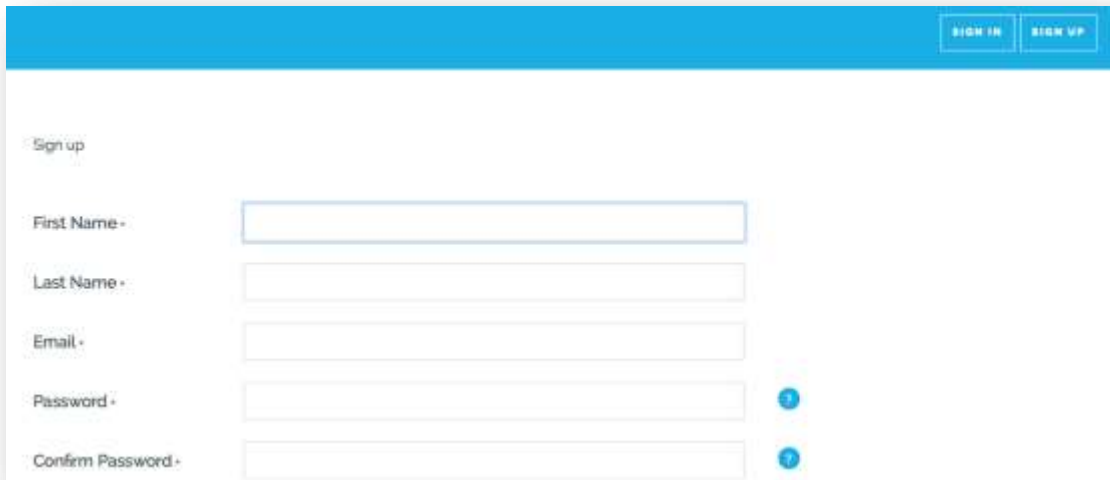
³ Required for recording

Obtaining and Using a Vendor Key

Obtaining a Vendor Key

Each Agora account can create multiple projects, and each project has a unique vendor key.

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

A screenshot of the Agora dashboard sign-up form. The form is titled "Sign up" and includes fields for "First Name", "Last Name", "Email", "Password", and "Confirm Password". There are "SIGN IN" and "SIGN UP" buttons at the top right. The "Password" and "Confirm Password" fields have a small blue icon with a "1" next to them, indicating a password strength indicator.

2. Click **ADD NEW** on the **PROJECTS** page of the dashboard.
3. Fill in the **Project Name**. Skip the **Enable Sign Key** unless you plan to generate a dynamic key, and click **SAVE**.
4. Find your vendor key under the project that you have created.

A screenshot of the "Project Name" form. The "Project Name" field is highlighted with a red box and contains the text "TEST1". Below the field is a checkbox labeled "Enable Sign Key". At the bottom right are "CLOSE" and "SAVE" buttons.

Using a Vendor Key

Access the Agora services by using your unique vendor key:

Enter the vendor key in the web demo application (only for web-application).

Set the **key** parameter to the value of the vendor key when joining a session.

Obtaining and Using a Dynamic Key

Obtaining a Vendor Key and a Sign Key

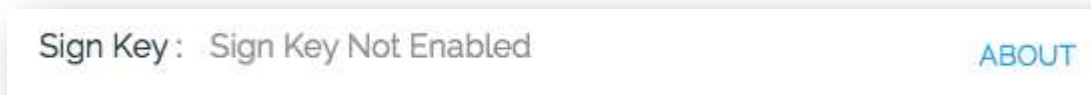
Each Agora account can create multiple projects, and each project has a unique vendor key and sign key. Follow the steps below to create a project to obtain a vendor key and a sign key at the same time:

1. Log in to <https://dashboard.agora.io>.
Click **ADD NEW** on the PROJECTS page of the dashboard. Fill in the Project Name, check the Enable Sign Key, and click **SAVE**.

A project is added.
2. Find the vendor and sign keys of the new project as follows:



3. Click the lock icon to show the sign key, and re-click it to hide the sign key again.
4. If you did not select **Enable Sign Key** when creating the project, click **ABOUT** to view the sign key after the project is created:



Notes

- If you want to renew a sign key, contact support@agora.io.
- Keep your sign key on the server and never on any client machine.
- See the table in [Dynamic Key Structure](#) for sign key use.

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. For the sample code, refer to either of the following sites:

- Github: <https://github.com/AgoraLab/AgoraDynamicKey>

Using a Dynamic Key

Before a user joins a session to access the Agora service:

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.
3. The dynamic key is based on the sign key, vendor key, Channel Name, Current Timestamp, Client User ID, Lifespan Timestamp, and so on.
4. Set the Key parameter to the value of the dynamic key when the client application joins a session.
5. 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.

Increased Security with Dynamic Keys

If your organization chooses to use dynamic keys, before a user joins a channel, the client application must provide a new dynamic key. The signaling server verifies each user identity. The dynamic key uses the HMAC/SHA1 signature schema to increase security for communications within your organization. The table below outlines the structure of the dynamic key. Connect all fields in the sequence shown.

Field	Type	Length	Description
Version	String	3	Dynamic key version information
Sign	String	40	<p>Hex code for the signature. A 40-byte string calculated by the HMAC algorithm based on inputs including the sign key and the following fields:</p> <ul style="list-style-type: none"> • Service Type: Visible ASCII string provided by Agora.io. See Service Type. • Vendor Key: 32-character 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, maximum length: 64-bytes. • User ID: The User ID defined by the client. • Call Expiration Timestamp: The timestamp indicates that from the specific moment the user cannot communicate in the channel 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. It authorizes access to the Agora service for 5 minutes.
Random Number	Integer	8	A 32-bit integer in hex code; generated upon each query.
Call Expiration Timestamp	Number	10	<p>Set the value to 0 for no limitation to the time of termination.</p> <p>Indicates the exact time when a party can no longer use the Agora service (for example, when someone is forced to leave an ongoing call).</p>

Service Type

Service	Value	Description
Session	ACS	The audio and video services provided by Agora.io. For example, when you call the joinChannel API, and a dynamic key is required, 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, PHP, Python, Ruby and others. For more information, refer to:

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

Realizing Whiteboard Function – Web Application

Deploying Your Web Application

Applications using the Agora Whiteboard SDK are standard JavaScript applications. To deploy the application, you need to load the Agora JS library and also need access to the JS extension libraries provided with the SDK. Follow your normal JavaScript hosting procedures when deploying the JavaScript libraries:

1. Load the Agora JS library:
AgoraWBSDK-1.0.0.js
2. JS extension libraries are also needed:
vendor-bundle.js

Running the Demo Application

Client

The default sample code only requires your static vendor key (which is entered into the demo application web page).

To run the provided demo application:

1. Ensure that you have a local web server installed, such as Apache, NginX, or Node.js.
2. Deploy the files under ./client/ to your web server, set the Key value in the file script-host.js and script-guest.js to your vendor key, and then launch your http/https service.
3. Access the demo application page on your web server using one of the browsers listed in Section [Supported Web Browsers](#).

Do the following if you want to run the demo application with a dynamic key:

Note: For more information, see [Obtaining and Using a Dynamic Key](#).

1. Set up and launch the key-generation server.
2. Make an http or https client request for the key-generation server to get a new dynamic key.
3. In script-host.js and script-guest.js (both under client/js/), replace the Key with the newly generated dynamic key before calling Agora.Whiteboard.join.

Server

This code is only needed if you want to experiment with using the more secure dynamic key. In production environment, you integrate this logic into your own server-side applications and (re)code this in the programming languages you are already using for your server-based functionality.

The sample code is in JavaScript and requires a standard Node.js server:

1. Install a standard Node.js server within your server or cloud infrastructure.
2. Run 'npm install' under ../server/nodejs/.
3. Fill in the values of your VENDOR_KEY and SIGN_KEY in ../server/nodejs/DemoServer.js.
4. Launch the server with 'node DemoServer.js'.

Realizing the Whiteboard Function – Non-Web Application

Non-Web application (on any of the following platforms, iOS, Android, Mac or Windows) developers can use WebView to access the Agora Whiteboard URL service:

- 🔗 **URL:** <https://wb.agora.io/html/agora-wb.html>
- 🔗 **Demo Usage:** <https://wb.agora.io/html/agora-wb.html?key=key1&cname=name1&role=host&uinfo=user1>

Parameter	Description	Required?
key	Key can be one of the following: <ul style="list-style-type: none">- vendor key: provided by Agora during registration.- dynamic key: the token generated with vendor key and sign key. A NodeJS implementation of token-gen algorithm is provided. This is the safest and recommended way to access the Agora Global Network.	Yes
cname	The unique channel name for the Agora whiteboard session.	Yes
role	User Role host : the organizer guest : the participant The organizer can control the whiteboard for all the activities, but the participant has view-only rights.	Yes
uinfo	The username to identify the user. If not specified, the server will generate a random user name.	No
expire	If the expiration date not specified, the validity period is 24 hours starting from the moment the channel is created.	No
width	The width of the whiteboard, and it is 1024 by default.	No
height	The height of the whiteboard, and it is 768 by default.	No

Agora Whiteboard SDK - API Reference

The Agora Whiteboard SDK library includes the following class:

Agora.Whiteboard	Use the Agora.Whiteboard object to join and leave shared sessions.
-------------------------	--

join(params)

This method allows user to join a shared session. It will create a session if the user initiates that call.

Parameter	Type	Description
params	Object	The only parameter for the join method, which includes the following settings.

key	String	Key can be one of the following: <ul style="list-style-type: none">- vendor key: provided by Agora during registration.- dynamic key: the token generated with vendor key and sign key. A NodeJS implementation of token-gen algorithm is provided. This is the safest and recommended way to access the Agora Global Network.
cname	String	The string that provides the unique channel name for the Agora.Whiteboard session.
host	String	The role of the user: <ul style="list-style-type: none">1: host0: attendee Host can take control of the whiteboard, attendee is view only.
expire	Number	If the expiration date not specified, the validity period is 24 hours starting from the moment the channel is created.
container	String	The identity of the DOM node.
width	String	The container width.
height	String	The container height
uinfo	String	(optional) user name

Sample code:

```
var hostParams = {  
    key      : 'f4637604af81440596a54254d53ade29',  
    cname    : 'PES-2017',  
    host     : 1,  
    width    : 800,  
    height   : 600,  
    container : "wbHost"  
};  
  
/* Call AgoraWhiteBoardApi */  
Agora.Whiteboard.join(hostParams);
```

leave(url)

This method allows users to leave a shared session.

Parameter Name	Type	Description
url	String	(optional) Specify the url that the user will be redirected to.

Sample Code:

```
Agora.Whiteboard.leave("http://sample.com");
```

