Zoom Electron SDK

Table of Contents

Latest SDK News

1. When creating the installer to install your SDK application on Windows, please add an additional step to run the command `cptinstall.exe -uninstall` with administrator privileges in the final stage of the installation (after all the related files of the SDK are copied successfully). This is to ensure that some users who have installed the old package can use the share function normally.

2. When publishing your app to Windows, please copy the Microsoft runtime libs to `bin` and `bin/aomhost` directories.

```
concrt140.dll
msvcp140.dll
msvcp140_1.dll
msvcp140_2.dll
msvcp140_codecvt_ids.dll
vccorlib140.dll
vcruntime140.dll
```

```
api-ms-win-core-console-l1-1-0.dll
api-ms-win-core-console-l1-2-0.dll
api-ms-win-core-datetime-l1-1-0.dll
api-ms-win-core-debug-l1-1-0.dll
api-ms-win-core-errorhandling-l1-1-0.dll
api-ms-win-core-file-l1-1-0.dll
api-ms-win-core-file-l1-2-0.dll
api-ms-win-core-file-l2-1-0.dll
api-ms-win-core-handle-l1-1-0.dll
api-ms-win-core-heap-l1-1-0.dll
api-ms-win-core-interlocked-l1-1-0.dll
api-ms-win-core-libraryloader-l1-1-0.dll
api-ms-win-core-localization-l1-2-0.dll
api-ms-win-core-memory-l1-1-0.dll
api-ms-win-core-namedpipe-l1-1-0.dll
api-ms-win-core-processenvironment-l1-1-0.dll
api-ms-win-core-processthreads-l1-1-0.dll
api-ms-win-core-processthreads-l1-1-1.dll
api-ms-win-core-profile-l1-1-0.dll
api-ms-win-core-rtlsupport-l1-1-0.dll
api-ms-win-core-string-l1-1-0.dll
api-ms-win-core-synch-l1-1-0.dll
api-ms-win-core-synch-l1-2-0.dll
api-ms-win-core-sysinfo-l1-1-0.dll
api-ms-win-core-timezone-l1-1-0.dll
api-ms-win-core-util-l1-1-0.dll
API-MS-Win-core-xstate-l2-1-0.dll
api-ms-win-crt-conio-l1-1-0.dll
api-ms-win-crt-convert-l1-1-0.dll
api-ms-win-crt-environment-l1-1-0.dll
api-ms-win-crt-filesystem-l1-1-0.dll
api-ms-win-crt-heap-l1-1-0.dll
api-ms-win-crt-locale-l1-1-0.dll
api-ms-win-crt-math-l1-1-0.dll
api-ms-win-crt-multibyte-l1-1-0.dll
api-ms-win-crt-private-l1-1-0.dll
api-ms-win-crt-process-l1-1-0.dll
api-ms-win-crt-runtime-l1-1-0.dll
api-ms-win-crt-stdio-l1-1-0.dll
api-ms-win-crt-string-l1-1-0.dll
api-ms-win-crt-time-l1-1-0.dll
api-ms-win-crt-utility-l1-1-0.dll
ucrtbase.dll
```

3. In version 5.2.42037.1112 of the Electron SDK, the support for Protocol Buffers is being added.

If you are building your own version of the Electron SDK, you will need to follow these steps:

- Download protobuf 3.4.0 source file and rename the src folder to protobuf_src.
- Copy the src folder into the lib/node_add_on folder.
- Run the build_nodeaddon script.

If you would like to use recent versions of protobuf(higher than 3.4.0), in addition to following the above steps, you must also do the following:

- Download the execution file of the corresponding protobuf and add its directory into the system path.
- In the terminal, navigate to the root directory of the Electron SDK(same level as the build_nodeaddon file).
- Run protoc.exe —js_out=import_style=common.js,binary:. lib/electron_sdk_proto command in the terminal to generate a electron_sdk_pb.js file. After generating this file, you will be able to use the interfaces provided by the Electron SDK.

If you are not building your own version of the Electron SDK and are using the Electron SDK provided by Zoom, this change will not impact your app and no further action is required on your end.

4. On macOS, SDK will verify the signature of all libraries. When the SDK libraries have been resigned, please call the interface `setTeamIdentifier` to set the organization unit of the signature before initializing in the app. For example:

```
    setTeamIdentifier(“the ou of certificate”);
```

```
Otherwise, some features, such as virtual background will not work after resigning the app.
```

5. Starting from 5.2.41735.0929, building the Electron SDK on Windows requires building with Visual Studio 2019.
6. Starting from Client SDK 5.0, if you are using tokens to start a meeting, you will only need to retrieve ZAK from Zoom API. The user token has been deprecated.
7. To follow with Zoom client's recent changes, Zoom SDK has temporary remove the "Unmute All" interface in Client SDK 5.0.
8. To align with Zoom’s recent announcement pertaining to our security initiative, Zoom Client SDKs have added **AES 256-bit GCM encryption** support, which provides more protection for meeting data and greater resistance to tampering. **The system-wide account enablement of AES 256-bit GCM encryption will take place on June 01.** You are **strongly recommended** to start the required upgrade to this latest version 4.6.21666.0428 at your earliest convenience. Please note that any Client SDK versions below 4.6.21666.0428 will **no longer be operational** from June 01.
9. We have merged and unified the `windows-electron-sdk` and the `mac-electron-sdk` into one single SDK.
   The new Electron SDK has a brand new structure, consist of the node-interface and the node-core:

- Node-interface: contains all the implementations by V8 engine
- Node-core: contains all the uniform interfaces for both Windows and Mac
  Due to the open source nature of this SDK, **you will be able to configure and compile the new Zoom Electron SDK with any versions of Electron.**

Community Support

You can find the community support forum here:

Community Forum

Disclaimer

Please be aware that all hard-coded variables and constants shown in the documentation and in the demo, such as Zoom Token, Zoom Access, Token, etc., are ONLY FOR DEMO AND TESTING PURPOSES. We STRONGLY DISCOURAGE the way of HARDCODING any Zoom Credentials (username, password, API Keys & secrets, SDK keys & secrets, etc.) or any Personal Identifiable Information (PII) inside your application. WE DON’T MAKE ANY COMMITMENTS ABOUT ANY LOSS CAUSED BY HARD-CODING CREDENTIALS OR SENSITIVE INFORMATION INSIDE YOUR APP WHEN DEVELOPING WITH OUR SDK.

Getting Started

The following instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

- If you need support or assistance, please visit our Zoom Developer Community Forum;

Prerequisites

Before you try out our SDK, you would need the following to get started:

- **A Zoom Account**: If you do not have one, you can sign up at https://zoom.us/signup.
  - Once you have your Zoom Account, sign up for a 60-days free trial at https://marketplace.zoom.us/
- **A device with Mac OS or Windows OS**:
  - Mac OS: MacOS 10.10 or later.
  - Windows: Windows 7 or later. Currently Windows 10 UWP is not supported.

Installing

Structure of Zoom Electron SDK

```
├── [sdk]
    ├── [mac] <-- Node file built by Zoom for mac
    ├── [win32] <-- Node file built by Zoom for win
├── binding.gyp
├── build_nodeaddon_mac.sh <-- use to rebuild node file for mac
├── build_nodeaddon_win_ia32.bat <-- use to rebuild node file for win
├── readme.txt / readme.md
├── run_demo_mac.sh
├── run_demo_win.bat <-- use to run demo for win
├── [demo] <-- demo app is inside
└── [lib] <-- js files and source code of Zoom Electron SDK
build_nodeaddon_mac.sh / build_nodeaddon_win_ia32.bat
```

Development environment configuration for Windows

```
Note that Windows electron add-on is 32bit.
```

1. Install electron and node.js

   - how to install node.js 12.16.1 version，download url: https://nodejs.org/download/release/v12.16.1/
   - install electron 8.2.4 version,use command run `npm install --arch=ia32 --save-dev electron@8.2.4 -g`

2. run `npm install node-gyp -g` to install node-gyp

3. run `npm install bindings -g` to install bindings

4. make sure you installed msvc-2019 and python 2.7

5. `npm config set msvs_version 2019`
   `npm config set python python2.7`
   `npm config set npm_config_arch ia32`
   `npm config set npm_config_target_arch ia32`

Development environment configuration for Mac

1. Install node.js 12.16.1 version，download url: https://nodejs.org/download/release/v12.16.1/.
   also can run `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"` and `sudo brew install node` to install node.js

2. Install electron 8.2.4 version，use command run `npm install --save-dev electron@8.2.4 -g`

3. run `npm install node-gyp -g` to install node-gyp

4. run `npm install bindings -g` to install bindings

Running Zoom Electron SDK demo

If you would like to run the demo app:

For Windows

1. Navigate to the root directory of the SDK package (same level as this README file)
2. Run `npm install` to install the required dependencies to run the demo app
3. Run `run_demo_win.bat` to run the demo app

For macOS

1. Navigate to the root directory of the SDK package (same level as this README file)
2. Run `npm install` to install the required dependencies to run the demo app
3. Run `run_demo_mac.sh` to run the demo app

Note: Due to the macOS security mechanism, the demo app will be blocked from launching the first time, please go to System Preference > Security & Privacy > General > Allow running the demo app.

Rebuilding the zoom node file

We recommend you to **REBUILD** the zoom node file on your own machine because the Electron version you use may not be the same as Zoom does.

Due to the open source nature of this SDK, you will be able to configure and compile the new Zoom Electron SDK with any versions of Electron.

If you are building your own version of the Electron SDK, you will need to follow these steps:

- Download the `protobuf 3.4.0` source file and rename the `src` folder to `protobuf_src`.
- Copy the src folder into the `lib/node_add_on` folder.
- Run the `build_nodeaddon` script:
    - on Windows, please run the `build_noodeaddon_win_ia32.bat`
    - on macOS, please run the `build_nodeaddon_mac.sh`

If you would like to use recent versions of protobuf(higher than 3.4.0), in addition to following the above steps, you must also do the following:

- Download the execution file of the corresponding protobuf and add its directory into the system path.
- In the terminal, navigate to the root directory of the Electron SDK(same level as the build_nodeaddon file).
    - Run `protoc.exe –js_out=import_style=common.js,binary:. lib/electron_sdk_proto` command in the terminal to generate an electron_sdk_pb.js file. After generating this file, you will be able to use the interfaces provided by the Electron SDK.

Initializing SDK with JWT token

When initializing the SDK, you will need to compose a JWT token using your SDK key & secret.

- How to compose JWT token for SDK initialization

You may generate your JWT token using the online tool https://jwt.io/. **It is highly recommended to generate your JWT token in your backend server.**

JWT is generated with three core parts: Header, Payload, and Signature. When combined, these parts are separated by a period to form a token: aaaaa.bbbbb.cccc.

Please follow this template to compose your payload for SDK initialization:

- Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

- Payload

```
{
         "appKey": "string",     // Your SDK key
         "iat": long,   // access token issue timestamp (unit: second)
         "exp": long,  // access token expire timestamp, MAX: iat + 2 days (unit: second)
         "tokenExp": long // token expire timestamp, MIN:iat + 30 minutes (unit: second)
}
```

The minimum value of `tokenExp` should be at least 30 minutes, otherwise, SDK will reject the authentication request.

- Signature

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  "Your SDK secret here"
)
```

You do not need to secret base64 encoded your signature. Once the JWT token is generated, please do not reveal it or publish it. **It is highly recommended to handle your SDK key and secret and generate JWT in a backend server to be consumed by your application. Do not generate JWT in a production application.**

SDK Reference

A local copy of the SDK reference is also included with this package.

Change log

Please refer to our CHANGELOG.pdf for all changes.

Frequently Asked Questions (FAQ)

1. `How to sign Electron SDK app on MacOS? Why my Electron SDK app crashes on MacOS after signing?`:
   - You may use the following command line to sign Electron SDK app on MacOS:

```
codesign --force --verify --verbose --entitlements runtime.entitlements --options runtime --sign "Developer
```

Please note that: You MUST use runtime entitlement to sign your Electron SDK on MacOS, and the entitlement MUST include the permission to use "Audio Input" and "Camera", otherwise, the app will crash due to Apple's privacy violation.

- Not finding what you want? We are here to help! Please visit our Zoom Developer Community Forum for further assistance.

Support

For any issues regarding our SDK, please visit our new Community Support Forum at https://devforum.zoom.us/.

License

Use of this software is subject to important terms and conditions as set forth in the License file

Please refer to LICENSE.pdf file for details
