


audacity / audacity Public

[Code](#) [Issues 766](#) [Pull requests 127](#) [Discussions](#) [Actions](#) [Projects 11](#) [Wiki](#) [Security](#) [Insights](#)

master ▾

...

audacity / BUILDING.md

 **crsib** Documentation is updated to reflect the updates to the build system History

8 contributors



320 lines (210 sloc) 11.5 KB

...

# Building Audacity

## Prerequisites

- `python3`  $\geq 3.8$
- `cmake`  $\geq 3.16$
- A working C++ 17 compiler
- Graphviz (optional)

For Windows see below for important installer settings.

Please note that Xcode 14 support on macOS requires CMake 3.24.0 or later.

## Conan

Audacity uses [Conan](#) to manage dependencies. If `conan` is not already installed, Audacity will download and install it automatically.

However, if you want to install Conan manually, you can do so by following the instructions on the [Conan website](#). Manual installation can be useful if you want to use Conan to manage dependencies for other projects or if you plan to have multiple builds of Audacity on the same machine.

## CMake

On Windows, please use the [prebuilt binaries](#). Ensure you select one of the options to add cmake to the system path.

On macOS, the easiest way to install CMake is `brew install cmake`.

On Linux, `cmake` is usually available from the system package manager. Alternatively, `sudo pip3 install cmake` can be used to install the latest version of CMake.

## Windows

We build Audacity using [Microsoft Visual Studio](#) 2019 and 2022. In order to build Audacity **Desktop development with C++** workload is required.

## macOS

We build Audacity using XCode versions 12 and later. However, it is likely possible to build it with XCode 7.

## Linux

We use GCC 9 and later, but any C++17 compliant compiler should work.

Here are the dependencies you need to install on various distribution families.

### Debian and Ubuntu

```
$ sudo apt-get update
$ sudo apt-get install -y build-essential cmake git python3-pip
$ sudo pip3 install conan
$ sudo apt-get install libgtk2.0-dev libasound2-dev libjack-jackd2-dev uuid-dev
```

### openSUSE

```
$ sudo zypper refresh
$ sudo zypper install patterns-devel-C-C++-devel_C_C++ cmake git python3-pip \
    gtk2-devel libjack-devel uuid-devel libSM-devel
$ sudo pip3 install conan
```

### Fedora Workstation

```
$ sudo dnf update
$ sudo dnf install gcc-c++ cmake git python3-pip perl-core \
    gtk2-devel gtk3-devel alsa-lib-devel jack-audio-connection-kit-devel uuid-devel libSM-devel
$ sudo pip3 install conan
```

## Graphviz

<https://graphviz.org/download/>

This is not necessary to build and run Audacity. It generates diagrams that aid understanding of the large scale structure of Audacity's source code.

If you install Graphviz, then an image file `modules.dot.svg` is generated in the build directory as a by-product of configuration. It shows the dependencies among the Audacity executable, its optional extension modules, its shared libraries, and third-party libraries.

You will also be able to change to the scripts directory and run `./graph.pl` to generate a diagram of dependencies among source code files within the executable.

## Building on Windows

1. Ensure the Python installer option `Add Python 3.x to PATH` is checked. Go to Windows Settings "Add or Remove Programs" and check the `Add Python to environment variables` in Python settings if Python is not in `PATH`.
2. Clone Audacity from the Audacity GitHub project.

For example, in the `git-bash` run:

```
$ git clone https://github.com/audacity/audacity/
```

### 3. Open CMake GUI.

Set the **Where is the source code** to the location where Audacity was cloned.

Set **Where to build the binaries** to the location you want to place your build in. It is preferred that this location is not within the directory with the source code.

### 4. Press **Configure**. You can choose which version of Visual Studio to use and the platform to build for in the pop-up. We support **x64** and **Win32** platforms. The **x64** platform is a default option. Press **Finish** to start the configuration process.

### 5. After successful configuration, you will see `Configuring done` in the last line of the log. Press **Generate** to generate the Visual Studio project.

### 6. After you see "Generating done", press **Open Project** to open the project in Visual Studio.

### 7. Select "Build -> Build Solution".

### 8. You can now run and debug Audacity!

Generally, steps 1-5 are only needed the first-time you configure. Then, after you've generated the solution, you can open it in Visual Studio next time. If the project configuration has changed, the IDE will invoke CMake internally.

## Building with ASIO support on Windows

To enable ASIO support, please select `audacity_has_asio_support=On` in CMake after the initial configuration and then run select **Configure** again as described above. ASIO is only supported on Windows and only for 64-bit builds.

## macOS

### 1. Clone Audacity from the Audacity GitHub project.

```
$ git clone https://github.com/audacity/audacity/
```

### 2. Configure Audacity using CMake:

```
$ mkdir build && cd build  
$ cmake -GXcode ../audacity
```

### 3. Open Audacity XCode project:

```
$ open Audacity.xcodeproj
```

and build Audacity using the IDE.

Steps 1 and 2 are only required for first-time builds.

Alternatively, you can use **CLion**. If you chose to do so, open the directory where you have cloned Audacity using CLion and you are good to go.

Before Audacity 3.2 only **x86\_64** builds and XCode "legacy" build system were supported. In order to build older version please use:

```
$ mkdir build && cd build  
$ cmake -GXcode -T buildsystem=1 ../audacity
```

to configure Audacity.

## Linux & Other OS

---

1. Clone Audacity from the Audacity GitHub project.

```
$ git clone https://github.com/audacity/audacity/
```

2. Configure Audacity using CMake:

```
$ mkdir build && cd build  
$ cmake -G "Unix Makefiles" ../audacity
```

By default, Debug build will be configured. To change that, pass `-DCMAKE_BUILD_TYPE=Release` to CMake.

3. Build Audacity:

```
$ make -j`nproc`
```

4. Testing the build: Adding a "Portable Settings" folder allows Audacity to ignore the settings of any existing Audacity installation.

```
$ cd Debug/bin  
$ mkdir "Portable Settings"  
$ ../audacity
```

5. Installing Audacity

```
$ cd <build directory>  
$ sudo make install
```

## Advanced

---

### CMake options

You can use `cmake -LH` to get a list of the options available (or use CMake GUI or `ccmake`). The list will include documentation about each option.

### Building using system libraries

On Linux it is possible to build Audacity using (almost) only the libraries provided by the package manager. Please, see the list of required libraries [here](#).

```
$ mkdir build && cd build  
$ cmake -G "Unix Makefiles" \  
    -Daudacity_use_ffmpeg=loaded \  
    -Daudacity_lib_preference=system \  
    -Daudacity_obey_system_dependencies=On \  
    ../audacity
```

There are a few cases when the local library build is preferred:

1. **wxWidgets**: While Audacity on **Linux** uses vanilla version of wxWidgets, we **require** that version **3.1.3** is used. This version is not available in most of the distributions.
2. **portaudio-v19**: Audacity currently uses **some private APIs**, so using system portaudio is not yet possible.
3. **vamp-host-sdk**: Development packages are not available in Ubuntu 20.04.
4. **libnyquist & portmixer**: Libraries are not available in Ubuntu 20.04.
5. **sqlite3 & libsmb**: Libraries are very outdated in Ubuntu 20.04.

It is not advised to mix system and local libraries, except for the list above. **zlib** is a very common dependency; it is possible to mix system and local libraries in one build. However, we advise against doing so.

There is a **Dockerfile** that can be used as an example of how to build Audacity using system libraries:

```
$ docker build -t audacity_linux_env .\linux\build-environment\
$ docker run --rm -v ${pwd}:/audacity/audacity/ -v ${pwd}/../build/linux-system:/audacity/build -it
audacity_linux_env
```

To find system packages, we rely on **pkg-config**. There are several packages that have broken **\*.pc** or do not use **pkg-config** at all. For the docker image - we handle this issue by installing the correct **pc files**.

## Disabling Conan

Conan can be disabled completely using **-Daudacity\_conan\_enabled=Off** during the configuration. This option implies **-Daudacity\_obey\_system\_dependencies=On** and disables **local** for packages that are managed with Conan.

## Disabling pre-built binaries downloads for Conan

It is possible to force Conan to build all the dependencies from the source code without using the pre-built binaries. To do so, please pass **-Daudacity\_conan\_allow\_prebuilt\_binaries=Off** to CMake during the configuration.

Additionally, passing **-Daudacity\_conan\_force\_build\_dependencies=On** will force Conan to rebuild all the packaged during every configuration. This can be useful for the offline builds against the Conan download cache.

## Troubleshooting Conan issues

To fix errors, similar to the following:

```
ERROR: HTTPSConnectionPool(host='center.conan.io', port=443): Max retries exceeded with url: /v1/ping (Caused by
SSLError(SSLCertVerificationError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: certificate has
expired (_ssl.c:1131)')))
```

please, update your Conan version. Alternatively, the problem can be fixed using:

```
$ conan config install https://github.com/conan-io/conanclientcert.git
```

For errors like:

```
Unable to connect to conan-center=https://conan.bintray.com
```

please run

```
conan remote remove conan-center
```

For errors like:

```
[package] package_name/package_version: package has 'exports_sources' but sources not found
```

please run

```
conan remove package_name/package_version -f
```

## Reducing Conan cache size

In order to reduce the space used by Conan cache, please run:

```
$ conan remove "*" --src --builds --force
```

## Selecting target architecture on macOS

Starting with version 3.2.0, Audacity target architecture on macOS can be selected by passing `MACOS_ARCHITECTURE` to the CMake during the configuration.

To build for Intel:

```
$ cmake -GXcode -DMACOS_ARCHITECTURE=x86_64 ../audacity
```

To build for AppleSilicon:

```
$ cmake -GXcode -DMACOS_ARCHITECTURE=arm64 ../audacity
```

The default build architecture is selected based on `CMAKE_HOST_SYSTEM_PROCESSOR` value.

When cross-compiling from Intel to AppleSilicon, or if *Rosetta 2* is not installed on the AppleSilicon Mac, a native Audacity version build directory is required, as Audacity needs a working `image-compiler`.

For example, to build ARM64 version of Audacity on Intel Mac:

```
$ mkdir build.x64
$ cmake -GXcode -DMACOS_ARCHITECTURE=x86_64 -B build.x64 -S ../audacity
$ cmake --build build.x64 --config Release --target image-compiler
$ mkdir build.arm64
$ cmake -GXcode -DMACOS_ARCHITECTURE=arm64 -DIMAGE_COMPILER_EXECUTABLE=build.x64/Utils/RelWithDebInfo/image-compiler
-B build.arm64 -S ../audacity
$ cmake --build build.arm64 --config Release
```

This will place ARM64 version into `build.arm64/Release/`.

## Building with VST3SDK without Conan (Linux only)

Set one of the following environment variables to the path to the VST3 SDK (i.e. the folder containing the `plugininterfaces` folder):

- `VST3_SDK_DIR`
- `VST3SDK_PATH`
- `VST3SDK`

or copy the VST3 SDK to `vst3sdk` directory in the Audacity source tree.

Pass `-Daudacity_use_vst3sdk=system` to CMake. CMake will build the SDK during the configuration.

[Give feedback](#)