

Android audio architecture

Android's audio Hardware Abstraction Layer (HAL) connects the higher-level, audio-specific framework APIs in [android.media](#) to the underlying audio driver and hardware. This section includes implementation instructions and tips for improving performance.

Android audio architecture defines how audio functionality is implemented and points to the relevant source code involved in the implementation.

Application framework

The application framework includes the app code, which uses the [android.media](#) APIs to interact with audio hardware. Internally, this code calls corresponding JNI glue classes to access the native code that interacts with audio hardware.

JNI

The JNI code associated with [android.media](#) calls lower level native code to access audio hardware. JNI is located in `frameworks/base/core/jni/` and `frameworks/base/media/jni`.

Native framework

The native framework provides a native equivalent to the [android.media](#) package, calling Binder IPC proxies to access the audio-specific services of the media server. Native framework code is located in `frameworks/av/media/libmedia`.

Binder IPC

Binder IPC proxies facilitate communication over process boundaries. Proxies are located in `frameworks/av/media/libmedia` and begin with the letter "I".

Media server

The media server contains audio services, which are the actual code that interacts with your HAL implementations. The media server is located in `frameworks/av/services/audioflinger`.

HAL

The HAL defines the standard interface that audio services call into and that you must implement for your audio hardware to function correctly. For more details, refer to the [audio HAL interface](#) and comments in the *.hal files of the corresponding HAL version directory.

Kernel driver

The audio driver interacts with your hardware and HAL implementation. You can use Advanced Linux Sound Architecture (ALSA), Open Sound System (OSS), or a custom driver (HAL is driver-agnostic).

Note: If you use ALSA, we recommend **external/tinyalsa** for the user portion of the driver because of its compatible licensing (the standard user-mode library is GPL-licensed).

