Playing audio

People expect rich audio experiences that automatically adjust when the context changes on the device.

Devices can play audio in a variety of ways, such as through internal or external speakers, headphones, and wirelessly through devices that use Bluetooth or AirPlay. To manipulate sound on their devices people use several types of controls, including volume buttons, the Ring/Silent switch on iPhone, headphone controls, the Control Center volume slider, and sound controls in third-party accessories. Whether sound is a primary part of your experience or a supporting element, you need to make sure it behaves as people expect as they make changes to volume and output.

Silence. People switch a device to silent when they want to avoid being interrupted by unexpected sounds like ringtones and incoming message tones. In this scenario, they also want to silence nonessential sounds, such as keyboard clicks, sound effects, game soundtracks, and other audible feedback. When a device is in silent mode, it plays only the audio that people explicitly initiate, like media playback, alarms, and audio/video messaging.

Volume. People expect their volume settings to affect all sound in the system — including music and in-app sound effects — regardless of the method they use to adjust the volume. An exception is the ringer volume on iPhone, which people can adjust separately in Settings.

Headphones. People use headphones to keep their listening private and in some cases to free their hands. When connecting headphones, people expect sound to reroute automatically without interruption; when disconnecting headphones, they expect playback to pause immediately.

Best practices

Adjust levels automatically when necessary — don't adjust the overall volume. Your app can adjust relative, independent volume levels to achieve a great mix of audio, but the system volume always governs the final output.

Permit rerouting of audio when possible. People often want to select a different audio output device. For example, they may want to listen to music through their living room stereo, car radio, or Apple TV. Support this capability unless there's a compelling reason not to.

Use the system-provided volume view to let people make audio adjustments. The volume view includes a volume-level slider and a control for rerouting audio output. You can customize the appearance of the slider. For developer guidance, see `MPVolumeView`.

Choose an audio category that fits the way your app or game uses sound. Depending on the audio category you choose, your app's sounds can mix with other audio, play while your app is in the background, or stop when people set the Ring/Silent switch to silent. As much as possible, pick a category that helps your app meet people's expectations. For example, don't make people stop listening to music from another app if you don't need to. For developer guidance, see `AVAudioSession.Category`.

Category	Meaning	Behavior
Solo ambient	Sound isn't essential, but it silences other audio. For example, a game with a soundtrack.	Responds to the silence switch. Doesn't mix with other sounds. Doesn't play in the background.
Ambient	Sound isn't essential, and it doesn't silence other audio. For examples, a game that lets people play music from another app during gameplay in place of the game's soundtrack.	Responds to the silence switch. Mixes with other sounds. Doesn't play in the background.
Playback	Sound is essential and might mix with other audio. For example, an audiobook or educational app that teaches a foreign language, which people might want to listen to after leaving the app.	Doesn't respond to the silence switch. May or may not mix with other sounds. Can play in the background.
Record	Sound is recorded. For example, a note-taking app that records audio memos or phone calls. An app of this nature might switch its category to playback if it lets people play the recorded notes.	Doesn't respond to the silence switch. Doesn't mix with other sounds. Can record in the background.
Play and record	Sound is recorded and played, potentially simultaneously. For example, an audio messaging or video calling app.	Doesn't respond to the silence switch. May or may not mix with other sounds. Can record and play in the background.

Respond to audio controls only when it makes sense. People can control audio playback from outside your app's interface — such as in Control Center or with controls on their headphones — regardless of whether your app is in the foreground or background. If your app is actively playing audio, in a clear audio-related context, or connected to a device that uses Bluetooth or AirPlay, it's fine to respond to audio controls. Otherwise, when people activate a control, avoid halting audio currently playing from another app.

Avoid repurposing audio controls. People expect audio controls to behave consistently in all apps, so it's essential to avoid redefining the meaning of an audio control in your app. If your app doesn't support certain controls, don't respond to them.

Consider creating custom audio player controls only if you need to offer commands that the system doesn't support. For example, you might want to define custom increments for skipping forward or backward, or present content that's related to the playing audio, such as a sports score.

Let other apps know when your app finishes playing temporary audio. If your app can temporarily interrupt the audio of other apps, be sure to fix your audio session in a way that lets them know when they can resume. For developer guidance, see `notifyOthersOnDeactivation`.

Handling interruptions

Although most apps and games rely on the system's default interruption behavior, you can customize this behavior to better accommodate your needs.

Determine how to respond to audio-session interruptions. For example, if your app supports recording or other audio-related tasks that people don't want interrupted, you can tell the system to avoid interrupting the currently playing audio in favor of allowing calls people choose to accept it. Another example is a VoIP app, which must end a call when people close the Smart Folio of their iPad while they're using the built-in microphone. Closing the Smart Folio automatically mutes the iPad microphone and by default interrupts the audio session associated with it. If a VoIP app restarts the audio session when people reopen their Smart Folio, it risks invading people's privacy by unmuting the microphone without their knowledge. You can inspect an audio-session interruption to help determine the right way to respond; for developer guidance, see Handling audio interruptions.

When an interruption ends, determine whether to resume audio playback automatically. Sometimes, audio from a different app can interrupt the audio your app is playing. An interruption can be resumable, like an incoming phone call, or nonresumable, like when people start a new music playback. Use the interruption type and your app's type to decide whether to resume playback automatically. For example, a media playback app that's actively playing audio when an interruption occurs can check to be sure the type is resumable before continuing playback when the interruption ends. On the other hand, a game doesn't need to check the interruption type before automatically resuming playback, because a game plays audio without an explicit user choice. For developer guidance, see `shouldResume`.

Platform considerations

iOS, iPadOS

Use the system's sound services to play short sounds and vibrations. For developer guidance, see Audio Services.

macOS

In macOS, notification sounds mix with other audio by default.

tvOS

In tvOS, the system plays audio only when people initiate it, through interactions within apps and games or when performing device calibrations. For example, tvOS doesn't play sounds to accompany components like alerts or notifications.

visionOS

Subtle, expressive sounds are everywhere in visionOS, enhancing experiences and providing essential feedback when people look at a virtual object and use gestures to interact with it. The system combines audio algorithms with information about a person's physical surroundings to produce Spatial Audio, which is sound that people can perceive as coming from specific locations in space, not just from speakers.

> **Important**
> In visionOS, as in every platform, avoid communicating important information using only sound. Always provide additional ways to help people understand your app. For guidance, see Accessibility.

In visionOS, audio playback from the Now Playing app pauses automatically when people close the app's window, and audio from an app that isn't the Now Playing app can duck when people look away from it to different app.

Prefer playing sound. People generally choose to keep sounds audible while they're wearing the device, so an app that doesn't play sound — especially in an immersive moment — can feel lifeless and may even seem broken. Throughout the design process, look for opportunities to create meaningful sounds that aid navigation and help people understand the spatial qualities of your app.

Design custom sounds for custom UI elements. In general, a system-provided element plays sound to help people locate it and receive feedback when they interact with it. To help people interact with your custom elements, design sounds that provide feedback and enhance the spatial experience of your app.

Use Spatial Audio to create an intuitive, engaging experience. Because people can perceive Spatial Audio as coming from anywhere around them, it works especially well in a fully immersive context as a way to help an experience feel lifelike. Another option provides pervasive sounds that can help place people in a virtual world and an audio source can sound like it comes from a specific object. As you build the soundscape for your app, consider using both types of audio.

Consider defining a range of places from which your app sounds can originate. Spatial Audio helps people locate the object that's making sound, whether it's stationary or moving in space. For example, when people move an app window that's playing audio, the sound continues to come directly from the window, wherever people move it.

Consider varying sounds that people could perceive as repetitive over time. For example, the system subtly varies the pitch and volume of the virtual keyboard's sounds, suggesting the different sounds a physical keyboard can make as people naturally vary the speed and forcefulness of their typing. An efficient way to achieve a pleasing variation in sound is to randomize a sound file's pitch and volume during playback, instead of creating different files.

Decide whether you need to play sound that's fixed to the wearer or tracked by the wearer. People perceive fixed sound as if it's pointed at them, regardless of the direction they look or the virtual objects they move. In contrast, people tend to perceive tracked sound as coming from a particular object, so moving the object closer or farther away changes what they hear. In general, you want to use tracked sound to enhance the realism of your experience, but there could be cases where fixed sound is a good choice. For example, MindBrows uses fixed sound to envelop the wearer in an engaging, peaceful setting.

watchOS

In watchOS, the system manages audio playback. An app can play short audio clips while it's active and running in the foreground, or it can play longer audio that continues even when people lower their wrist or switch to another app. For developer guidance, see Playing Background Audio.

Use the recommended encoding values for media assets. Specifically, use the 64 kbps HE-AAC (High-Efficiency Advanced Audio Coding) format to produce good-quality audio with lower data requirements.

Consider presenting a Now Playing view so people can control current or recently played audio without leaving your app. The system-provided Now Playing view also displays information about the current audio source — which might be another app on a person's Apple Watch or iPhone — and automatically selects the current or most recently used source. For developer guidance, see Adding a Now Playing View.

Resources

Related

Playing video

Feedback

Developer documentation

Configuring your app for media playback — AVFoundation

`AVAudioSession` — AVFAudio

Videos

Explore immersive sound design

Principles of spatial design

Immerse your app in Spatial Audio

Change log

Date	Changes
June 21, 2023	Updated to include guidance for visionOS.