

[AVFAudio](#) / AVAudioEngine

## Class

# AVAudioEngine

An object that manages a graph of audio nodes, controls playback, and configures real-time rendering constraints.

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 13.1+ | macOS 10.10+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

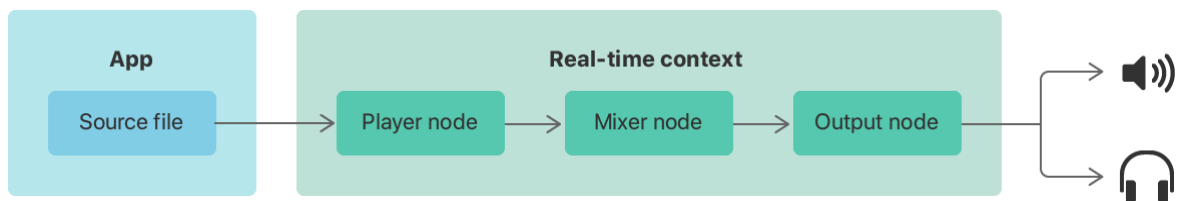
```
class AVAudioEngine
```

## Mentioned in

📄 Routing audio to specific devices in multidevice sessions

## Overview

An audio engine object contains a group of [AVAudioNode](#) instances that you attach to form an audio processing chain.



You can connect, disconnect, and remove audio nodes during runtime with minor limitations. Removing an audio node that has differing channel counts, or that's a mixer, can break the graph. Reconnect audio nodes only when they're upstream of a mixer.

By default, Audio Engine renders to a connected audio device in real time. You can configure the engine to operate in manual rendering mode when you need to render at, or faster than, real time. In that mode, the engine disconnects from audio devices and your app drives the rendering.

# Create an Engine for Audio File Playback

To play an audio file, you create an AVAudioFile with a file that's open for reading. Create an audio engine object and an AVAudioPlayerNode instance, and then attach the player node to the engine. Next, connect the player node to the audio engine's output node. The engine performs audio output through an output node, which is a singleton that the engine creates the first time you access it.

```
let audioFile = /* An AVAudioFile instance that points to file that's open for reading */
let audioEngine = AVAudioEngine()
let playerNode = AVAudioPlayerNode()

// Attach the player node to the audio engine.
audioEngine.attach(playerNode)

// Connect the player node to the output node.
audioEngine.connect(playerNode,
                    to: audioEngine.outputNode,
                    format: audioFile.processingFormat)
```

Then schedule the audio file for full playback. The callback notifies your app when playback completes.

```
playerNode.scheduleFile(audioFile,
                        at: nil,
                        completionCallbackType: .dataPlayedBack) { _ in
    /* Handle any work that's necessary after playback. */
}
```

Before you play the audio, start the engine.

```
do {
    try audioEngine.start()
    playerNode.play()
} catch {
    /* Handle the error. */
}
```

When you're done, stop the player and the engine.

```
playerNode.stop()
audioEngine.stop()
```

# Topics

## Creating an Audio Engine

```
init()
```

Creates an audio engine instance for rendering in real time.

## Attaching and Detaching Audio Nodes

```
func attach(AVAudioNode)
```

Attaches an audio node to the audio engine.

```
func detach(AVAudioNode)
```

Detaches an audio node from the audio engine.

```
var attachedNodes: Set<AVAudioNode>
```

A read-only set that contains the nodes you attach to the audio engine.

## Getting the Input, Output, and Main Mixer Nodes

```
var inputNode: AVAudioInputNode
```

The audio engine's singleton input audio node.

```
var outputNode: AVAudioOutputNode
```

The audio engine's singleton output audio node.

```
var mainMixerNode: AVAudioMixerNode
```

The audio engine's optional singleton main mixer node.

## Connecting and Disconnecting Audio Nodes

```
func connect(AVAudioNode, to: AVAudioNode, format: AVAudioFormat?)
```

Establishes a connection between two nodes.

```
func connect(AVAudioNode, to: AVAudioNode, fromBus: AVAudioNodeBus, to
Bus: AVAudioNodeBus, format: AVAudioFormat?)
```

Establishes a connection between two nodes, specifying the input and output busses.

```
func disconnectNodeInput(AVAudioNode)
```

Removes all input connections of the node.

```
func disconnectNodeInput(AVAudioNode, bus: AVAudioNodeBus)
```

Removes the input connection of a node on the specified bus.

```
func disconnectNodeOutput(AVAudioNode)
```

Removes all output connections of a node.

```
func disconnectNodeOutput(AVAudioNode, bus: AVAudioNodeBus)
```

Removes the output connection of a node on the specified bus.

## Managing MIDI Nodes

```
func connectMIDI(AVAudioNode, to: AVAudioNode, format: AVAudioFormat?,
eventListBlock: AUMIDIEventListBlock?)
```

Establishes a MIDI connection between two nodes.

```
func connectMIDI(AVAudioNode, to: [AVAudioNode], format: AVAudioFormat
?, eventListBlock: AUMIDIEventListBlock?)
```

Establishes a MIDI connection between a source node and multiple destination nodes.

```
func disconnectMIDI(AVAudioNode, from: AVAudioNode)
```

Removes a MIDI connection between two nodes.

```
func disconnectMIDI(AVAudioNode, from: [AVAudioNode])
```

Removes a MIDI connection between one source node and multiple destination nodes.

```
func disconnectMIDIInput(AVAudioNode)
```

Disconnects all input MIDI connections from a node.

```
func disconnectMIDIOutput(AVAudioNode)
```

Disconnects all output MIDI connections from a node.

```
func connectMIDI(AVAudioNode, to: AVAudioNode, format: AVAudioFormat?,
block: AUMIDIOutputEventBlock?)
```

Establishes a MIDI-only connection between two nodes.

Deprecated

```
func connectMIDI(AVAudioNode, to: [AVAudioNode], format: AVAudioFormat?, block: AUMIDIOutputEventBlock?)
```

Establishes a MIDI-only connection between a source node and multiple destination nodes.

Deprecated

## Playing Audio

```
func prepare()
```

Prepares the audio engine for starting.

```
func start() throws
```

Starts the audio engine.

```
var isRunning: Bool
```

A Boolean value that indicates whether the audio engine is running.

```
func pause()
```

Pauses the audio engine.

```
func stop()
```

Stops the audio engine and releases any previously prepared resources.

```
func reset()
```

Resets all audio nodes in the audio engine.

```
var musicSequence: MusicSequence?
```

The music sequence instance that you attach to the audio engine, if any.

## Manually Rendering an Audio Engine

```
func enableManualRenderingMode(AVAudioEngineManualRenderingMode, format: AVAudioFormat, maximumFrameCount: AVAudioFrameCount) throws
```

Sets the engine to operate in manual rendering mode with the render format and maximum frame count you specify.

```
func disableManualRenderingMode()
```

Sets the engine to render to or from an audio device.

```
func renderOffline(AVAudioFrameCount, to: AVAudioPCMBuffer) throws -> AVAudioEngineManualRenderingStatus
```

Makes a render call to the engine operating in the offline manual rendering mode.

## Getting Manual Rendering Properties

`typealias AVAudioEngineManualRenderingBlock`

The type that represents a block that renders the engine when operating in manual rendering mode.

`var manualRenderingBlock: AVAudioEngineManualRenderingBlock`

The block that renders the engine when operating in manual rendering mode.

`var manualRenderingFormat: AVAudioFormat`

The render format of the engine in manual rendering mode.

`var manualRenderingMaximumFrameCount: AVAudioFrameCount`

The maximum number of PCM sample frames the engine produces in any single render call in manual rendering mode.

`var manualRenderingMode: AVAudioEngineManualRenderingMode`

The manual rendering mode configured on the engine.

`var manualRenderingSampleTime: AVAudioFramePosition`

An indication of where the engine is on its render timeline in manual rendering mode.

`var isAutoShutdownEnabled: Bool`

A Boolean value that indicates whether autosutdown is in an enabled state.

`var isInManualRenderingMode: Bool`

A Boolean value that indicates whether the engine is operating in manual rendering mode.

## Using Connection Points

`class AVAudioConnectionPoint`

A representation of either a source or destination connection point in the audio engine.

`func connect(AVAudioNode, to: [AVAudioConnectionPoint], fromBus: AVAudioNodeBus, format: AVAudioFormat?)`

Establishes a connection between a source node and multiple destination nodes.

## Constants

`enum AVAudioEngineManualRenderingError`

Constants that describe error codes that the framework returns from manual rendering mode methods.

`enum AVAudioEngineManualRenderingMode`

The two modes for manual rendering.

`enum AVAudioEngineManualRenderingStatus`

Status codes that return from the render call to the engine operating in manual rendering mode.

## Instance Methods

```
func inputConnectionPoint(for: AVAudioNode, inputBus: AVAudioNodeBus) -> AVAudioConnectionPoint?
```

Returns connection information about a node's input bus.

```
func outputConnectionPoints(for: AVAudioNode, outputBus: AVAudioNodeBus) -> [AVAudioConnectionPoint]
```

Returns connection information about a node's output bus.

---

## Relationships

### Inherits From

`NSObject`

### Conforms To

`CVarArg`

`CustomDebugStringConvertible`

`CustomStringConvertible`

`Equatable`

`Hashable`

`NSObjectProtocol`