

[Audio Toolbox](#) / Analyzing audio performance with Instruments

Article

Analyzing audio performance with Instruments

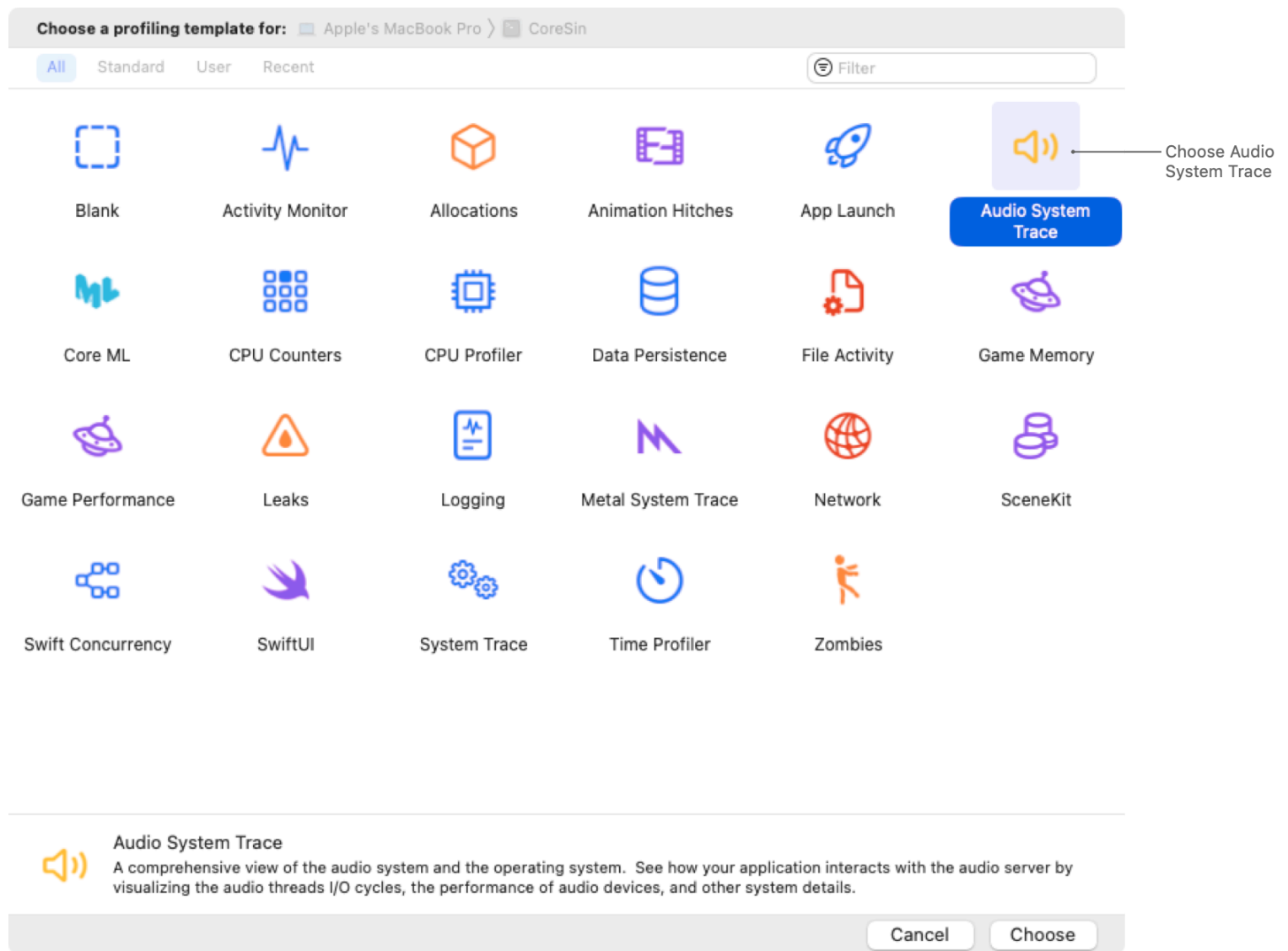
Ensure a smooth and immersive audio experience in your apps using Audio System Trace.

Overview

Poor quality audio playback can ruin the immersive experience in your app, making it crucial to maintain smooth audio playback by eliminating glitches and dropouts. When you encounter an audio glitch, you hear unintended distortion in the playback — pops, dropouts, and clicks. The Audio System Trace template includes several instruments that provide information about your app's performance and help you troubleshoot audio issues. The Audio Statistics, Audio Server, and Audio Client tracks are instruments in the Audio System Trace template that provide insights, errors, and warnings about the Core Audio system, and enable you to improve your app's audio experience.

Launch Audio System Trace

From Xcode's Product menu, choose Profile, or press Command-I. After Instruments launches, select Audio System Trace, then click Choose.



The Audio System Trace template includes the following instrument tracks:

Points of Interest

Indicates locations in the trace to which you may want to pay special attention.

System Load

Tracks the performance and current load of the system.

Thread State Trace

Tracks each time the operating system scheduler makes a decision that may impact your app's threads.

Virtual Memory Trace

Tracks virtual memory activity per thread.

System Call Trace

Records system calls and their duration.

Thermal State

Records the device's thermal state.

Audio Client

Tracks information about the timing of I/O process callbacks in your app.

Audio Statistics

Records engine jitter, which quantifies how late an I/O cycle was relative to its anticipated deadline and the number of concurrent audio threads.

Audio Server

Tracks engine timestamp and I/O cycle load and related points of interest.

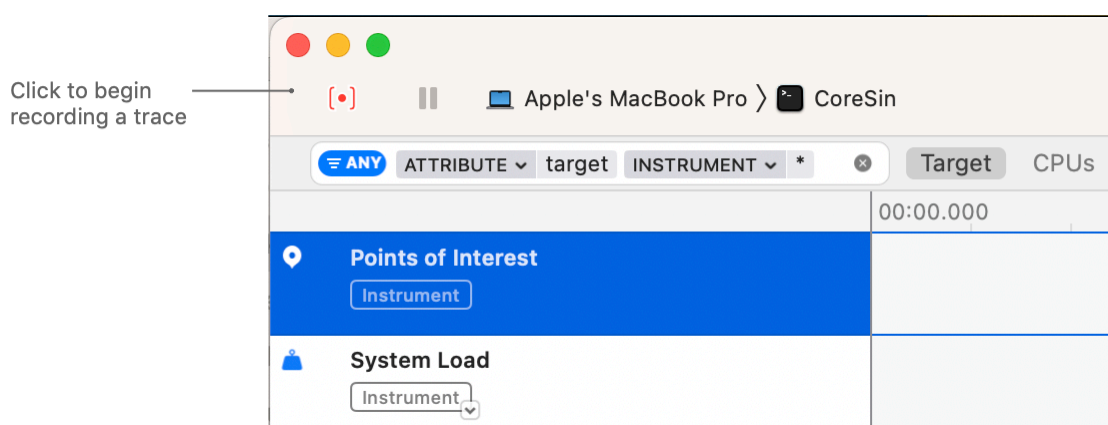
Hangs

Labels intervals by severity, measuring the duration of main thread blockage.

Explore the Audio System Trace instruments

Audio System Trace instruments include tracks to help you isolate where glitches are occurring and identify the cause of audio performance issues. You can view the audio system performance captured in an audio system trace in two ways:

- You can capture a trace by clicking the Record button in the toolbar. Your app launches Audio System Trace and starts recording. Within your app, perform the actions that reproduce the audio performance issue you want to analyze, and then click the Stop button to stop recording.
- You can open an audio system trace you previously recorded. In Instruments, choose File > Open, select a trace file, and click Open.



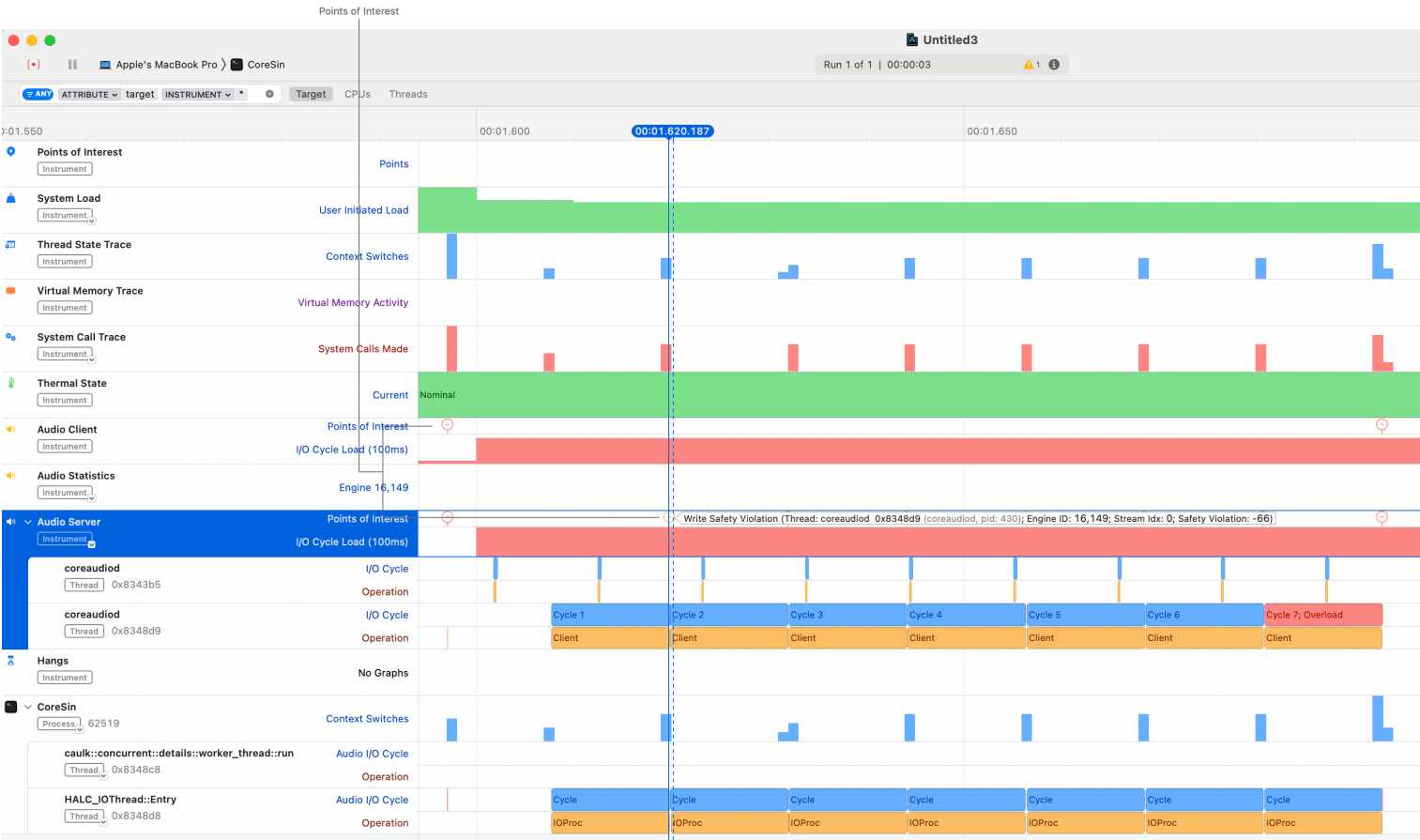
The audio system trace appears in a new window, along with the timeline and detail panes, as shown in the following screenshot.



Move your pointer along the top of the trace timeline pane to see points of interest along the Audio Client and Audio Server tracks. To identify audio performance issues, click the points of interest to view errors and warnings.



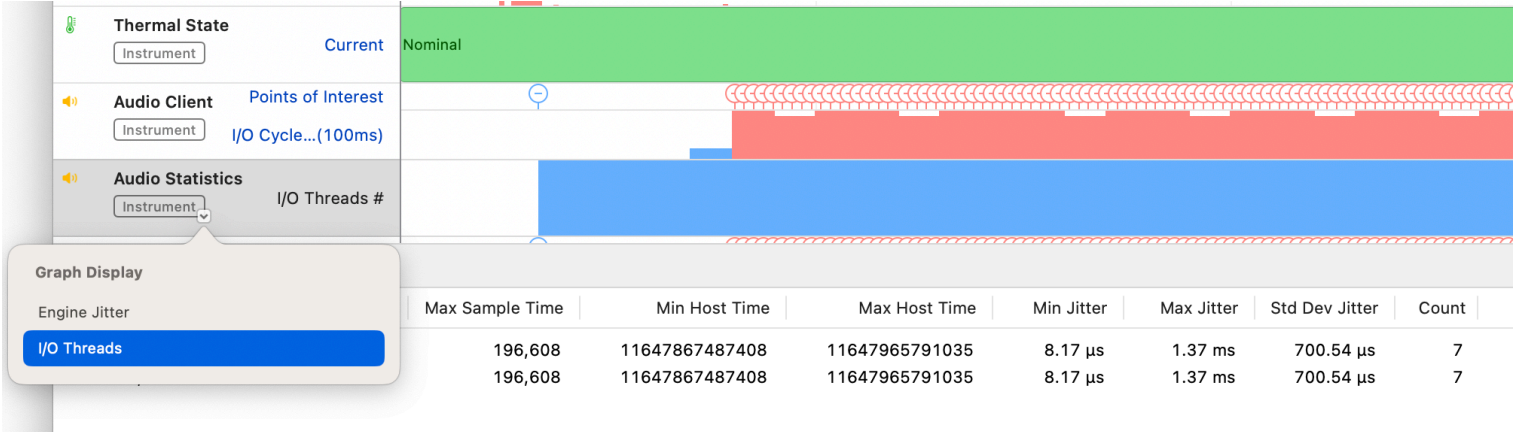
To have better granularity for a specific track, use your mouse or trackpad to zoom in to the corresponding area of the track for more information. Click a point of interest to update the detail pane for that selected track and view any corresponding detail information.



Analyze audio performance statistics

The Audio Statistics track gives an overview of each engine's jitter and sample time information to provide insights into the system's audio performance. The track displays two graphs: Engine Jitter and I/O Threads. The Engine Jitter graph quantifies deviations from an expected tick cadence. The I/O Threads graph shows the number of concurrent audio threads. The Engine represents a collection of one or more audio devices bound together.

Click the arrow next to the instrument name to switch between these two graphs.



The table below describes the colors and associated jitter times in the Audio Statistics track.

Color	Jitter time μ s (microseconds)
Green	0–30 μ s
Orange	31–100 μ s
Red	> 100 μ s

The Audio Statistics track’s detail pane includes the following menu items:

Engine ID

An identifier for an I/O entity within Audio Server.

Min/Max Sample Time

The shortest and longest sample times recorded for an engine.

Min/Max Host Time

The earliest and latest host timestamps for an engine.

Min/Max Jitter

The range of jitter values recorded for an engine.

Std Dev Jitter

The standard deviation from the mean jitter value during a recording.

Count

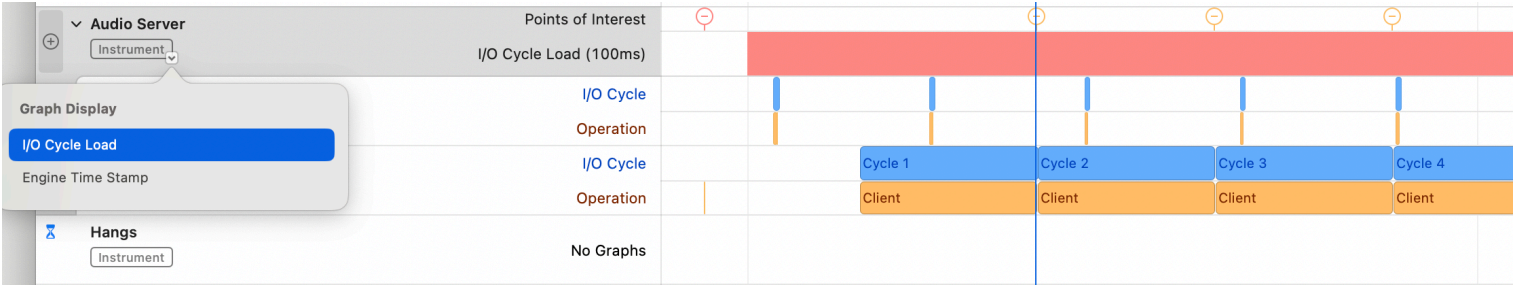
The number of events recorded in the run of the instrument tool.

Identify glitches with the Audio Server track

The Audio Server track provides Engine Time Stamp and I/O Cycle Load graphs, and related points of interest. Engine timestamp refers to the time when the audio engine processes audio, and it can help determine where a delay or other problem occurred in the audio processing chain. I/O cycle load refers to the time the system spends processing an audio buffer relative to the available time to process that buffer.

If the client doesn’t complete its operation during the allotted cycle time, the system can’t process the audio in real-time, leading to dropouts or glitches. If the I/O cycle load is consistently high, it indicates the audio processing might be too complex or intensive for the current buffer size and sample rate configuration.

Click the arrow next to the instrument name to switch between the Engine Time Stamp and I/O Cycle Load graphs.



As you view these tracks, pay particular attention to the red points of interest, which help you isolate where the audio issues occur. Typically, these issues occur when the I/O cycle takes too long. The points of interest include overloads, clock discontinuity errors, reanchors, and read safety violations.

An overload occurs if you don't deliver data on time, leading to issues in real-time audio playback. It's indicated in red on the graph and has a red point of interest. If you don't promptly write audio data to the output buffer, the speakers lack the necessary information for continuous playback and glitch until they receive the missing data. A clock discontinuity occurs when the clock has failed to stay in sync or there has been a break in the device's continuous samples. When detecting a clock discontinuity, Core Audio actively attempts to reanchor the device's clock with the timeline. A read safety violation tracks how close to a driver's safety offset a read was.

View zero timestamp information

Switching to the Zero Time Stamp detail pane in the Audio Server track can help you determine if an operation doesn't complete within an audio I/O cycle, leading to overloads and glitches.

A ring buffer is a fixed-size data structure where the last element connects to the first. Every time the audio driver completes a loop of the ring buffer, it generates a new zero timestamp. The following screenshot shows the Audio Server's detail pane with zero timestamp information.

Zero Time Stamp detail pane



The following menu items describe the zero timestamp information in the Audio Server track's detail pane shown above:

Sample Time

A running count of the total number of samples the driver has processed in the buffer.

Frame Count

The number of frames processed.

Host Time

The point when the buffer wraps.

Actual Host Ticks per Frame

The number of host ticks per audio frame.

Jitter

The deviation (in microseconds) from the expected timestamp.

Inspect the Audio Client track

The audio stack has a client/server architecture. To analyze audio system performance, you may want to examine what happens between the server and the client (your app). The Audio Client track provides information about the timing of I/O process callbacks for your app. Points of interest show issues with your app's audio performance and correlate to points of interest in the Audio Server track.

Use real-time operations

In the screenshot below, overloads are visible in the Audio Server track and are likely to cause glitches. Client operations might be conducting non-real-time operations that aren't safe, which can cause overloads. Blocking on a lock, allocating memory, or performing complex operations that exceed the allotted audio server time are examples of non-real-time operations that can cause glitches.

Complete multiple audio threads by a common deadline

Completing multiple real-time audio threads by a common deadline is important, or glitches can occur. An audio workgroup consists of real-time threads collaborating across various processes — such as the audio server, client apps, and plug-ins — to generate audio by a common deadline. The system uses audio workgroups to effectively monitor the CPU utilization of real-time threads, aiming to optimize the trade-off between energy efficiency and system performance.

Tip

Use Audio Workgroups API to optimize your audio threads' performance, giving the operating system insight into active real-time threads, including auxiliary ones that other apps and Audio Unit plug-ins created. For further insights into audio workgroups, see [Understanding Audio Workgroups](#).

Analyze backtrace information

With your app's track, you can select the detail row where the overload occurs to see the corresponding backtrace in the inspector pane, as shown in the screenshot below. The backtrace information can take you to your code that's causing audio performance issues.

Confirm tracks have no errors or warnings

When you've corrected all glitches appearing in your Audio Client and Audio Server tracks, you see the Audio Server track with the client audio completed within the time allotted by the server. As shown below, there are no errors or warnings in the tracks.

See Also

Utilities

≡ Audio Converter Services

Convert between linear PCM audio formats, and between linear PCM and compressed formats.

≡ Audio Session Support

Describe the properties that you associate with audio sessions and audio routes.

≡ Audio Toolbox Debugging

Obtain the internal state of Core Audio objects during the development and debugging of your code.

≡ Workgroup Management

Coordinate the activity of custom real-time audio threads with those of the system and other processes.

☰ Audio Codec
Translate audio data from one format to another.

☰ Clock Utilities
Manage time-related information associated with audio playback.