MusicReason RESOURCE

How MusicReason Tests Its Answers

https://musicreason.com/resources/how-musicreason-tests-its-answers/ · Last updated 2026-09-06

Trusting a result should not require taking it on faith. Every engine on this site is checked against a written set of fixtures, specific inputs paired with the exact output they must produce, before it ships, and every engine states its own limitations rather than implying it handles every case.

What a fixture actually is

Illustrative fixture types used across MusicReason, without exposing private source data

Fixture class	What it checks	Example
Golden case	The engine produces the theory-correct answer	C-E-G identifies as C major
Edge case	A boundary condition behaves correctly	An octave-exact interval does not misreport as a unison
Invalid case	A malformed or impossible input is rejected, not guessed at	An empty note selection returns "insufficient," not a chord
Invariant	A property that must always hold, across every input	Transposing up then down by the same interval returns the original note
Interpretation fixture	A genuinely ambiguous input produces the documented, non-arbitrary handling	A tied chord reading reports co-primary alternatives, not a coin flip

Deterministic engines: an exact match, every time

Chord Identifier and every other theory-based engine on this site is deterministic: the same input always produces the same output, so its fixtures require an exact match, not a close-enough approximation. A formula-based engine like Room Mode Calculator is checked the same way, against a worked numeric example computed independently from the published formula.

Recognition engines: synthetic signals with a known answer

Microphone Tuner and other signal-based engines are additionally checked against synthetic audio generated at a precisely known frequency, so the correct answer is known in advance rather than judged by ear. Edge fixtures deliberately include noisy, quiet, or ambiguous synthetic signals to confirm the engine reports lower confidence, or an alternative, rather than a falsely confident single answer. A representative-source stage using real recorded audio, not only synthetic signals, is required before any confidence-labeled claim is allowed to lock (see Why Audio Recognition Is Ambiguous).

Browser and privacy checks are part of "tested," too

An audio engine is not considered complete once its math passes. It is also checked for real in-browser behavior, mobile responsiveness, cancellation during an active analysis, and that no raw audio, filename, or detected result reaches analytics (Local Audio Processing and Privacy). A build that passes its math fixtures but fails a privacy check does not ship.

What happens when something needs to change

A locked engine or resource is not silently edited. A revision creates a new version and records the reason for the change, its scope, which routes and tests it affects, exactly what claim or method changed, the new validation evidence, and who signed off on it. The prior contract stays intact in the change history rather than disappearing.

How to read a stated limitation

When an engine states a limitation, for example that a chord tool works from pitch classes and cannot see your instrument's voicing, that is a scoped, deliberate boundary, not an apology for an unfinished feature. Passing every written fixture proves an engine behaves correctly on the specific cases it was checked against; it does not, and cannot, prove correctness for every conceivable input, which is exactly why the stated limitation exists alongside the passing test suite rather than instead of it.

Frequently asked questions

What is a "fixture" in MusicReason's testing?

A fixture is a specific input paired with the exact output the engine must produce, drawn from music theory conventions, published formulas, or a synthetic signal with a known answer. Every launch engine has a written fixture set it must reproduce exactly before it ships.

Does passing every fixture guarantee an engine is always right?

No. Fixtures prove the engine behaves correctly on the specific golden, edge, invalid, and invariant cases it was checked against. They cannot prove correctness for every possible input, which is why every recognition-style engine still carries an explicit confidence label and stated limitations rather than a claim of certainty.

What happens when a locked engine or resource needs to change?

A revision creates a new version and records the reason, scope, affected routes and tests, what claim or method changed, and the new validation evidence. The prior contract is never silently overwritten (Specs/status-and-locking.md).

Where this shows up

CHORDS AND HARMONY

Chord Identifier

ACOUSTICS

Room Mode Calculator

AUDIO ANALYSIS

Microphone Tuner

For the full policy narrative, see the Methodology page.