Harmony: Media-Carried Workflow Signals for Distributed AI Agent Execution

Milan Cheeks University of Colorado Boulder — M.S. in Computer Science program Independent Researcher
ORCID: 0009-0008-7507-0050 Corresponding author: mich3592@colorado.edu (primary) · milan@selflabs.xyz

Preprint — technical report. Not yet peer reviewed. The system described is patent pending (USPTO Provisional No. 63/587,817, converted to a nonprovisional application).

Abstract

AI agent systems are conventionally triggered through applications, APIs, or cloud consoles, while the media that people already consume — music, video, and ambient audio — initiates nothing. This paper presents Harmony, a system that embeds machine-readable workflow signals in the audio track of media and connects playback to approved agent work. An encoder compiles a compact digital packet (the HMY2 protocol) into a media file using near-ultrasonic frequency-shift keying. The packet carries no fixed task; it carries a routing identity consisting of an organization, a workflow, a channel, and an agent suite. An opt-in listening node decodes the packet locally, presents the cue to a runtime server, and the runtime assigns the next approved task from the organization's queue. Execution requires explicit user approval, and the result is recorded in a receipt. This decoupling of the media asset from the work it can initiate is the central contribution: a cue authored once can address a queue whose contents evolve over time, without re-encoding or redistributing the asset. We describe the complete prototype implementation — encoder studio, signal inspector, reliability bench, browser listening node, runtime server, and local executor — and we report its current validation status with the same rigor: the end-to-end loop has been demonstrated internally, while instrumented reliability measurement under real-world acoustic and codec conditions, production-grade authentication, and model-backed execution remain open work, for which we specify a five-part validation program with measured acceptance criteria.

Keywords: AI agents; audio watermarking; frequency-shift keying; workflow systems; human-in-the-loop execution; ambient computing

1. Introduction

The operational model of AI agents is converging on queues, tools, and approvals: work is defined in advance, an agent claims a task, executes it within scoped permissions, and produces a record. Yet the entry point to that model remains narrow. Every deployment begins with infrastructure — an application to open, an API to call, a console to log into. Meanwhile, an enormous, already-built distribution surface sits idle. Songs, videos, streams, and in-venue audio reach billions of playback events per day on devices that are fully capable of computation, and trigger nothing.

Harmony explores a simple question: can media itself serve as the addressing layer for agent work? The system embeds a small digital cue in a media file's audio. When a participating device plays or hears that media, an opt-in listening node decodes the cue and asks a runtime what work the cue addresses. The runtime assigns the next approved task from a queue. A user approves execution, an executor performs the task, and the system records a receipt.

Three design commitments shape the system. First, **the cue is a route, not a payload.** It names an organization, workflow, channel, and agent suite — comparable to a routing key in a workflow system — rather than carrying the task itself. The media asset is therefore stable while the work behind it changes. Second, **detection is local and listening is opt-in.** Raw audio never leaves the device; the runtime receives routing metadata and receipts, not recordings. Third, **execution is approval-gated.** A decoded cue can request work, but no agent runs without explicit user confirmation, and high-impact action classes are blocked in the prototype's action model.

This paper makes three contributions. (1) It defines the media-carried workflow cue: a two-packet-type protocol (static task packets and workflow cue packets) that separates media lifetime from task lifetime. (2) It presents a complete prototype stack implementing encode, inspect, listen, route, execute, and receipt as six working components. (3) It gives an unusually explicit account of what the prototype does and does not establish, together with a validation program designed to convert an internally demonstrated loop into a measured delivery channel.

Section 2 situates the work against audio watermarking, ultrasonic communication, and agent triggering. Section 3 states the design goals. Section 4 describes the signal layer, Section 5 the system architecture and implementation, and Section 6 the runtime and execution model. Section 7 analyzes security and privacy, including the prototype's known weaknesses. Section 8 reports evaluation status honestly and defines the validation program. Sections 9 and 10 discuss implications and limitations, and Section 11 concludes.

2. Background and Related Work

Digital audio watermarking. Embedding auxiliary data in audio has a long history in copyright protection and broadcast monitoring. Classical watermarking optimizes for imperceptibility and robustness against signal processing, typically at very low payload rates, and the data embedded is usually a static identifier or rights statement [1][2][3]. Harmony shares the low-rate, robustness-constrained channel but changes the semantics of the payload: the embedded data is not an identifier of the media, it is an address for work, interpreted by a runtime at playback time.

Ultrasonic and near-ultrasonic data channels. A separate line of work uses high-frequency audio as a short-range communication channel, and the security literature has studied it largely as a threat: ultrasonic side channels have been used for cross-device tracking, and inaudible or near-inaudible sound has been analyzed as an attack and exfiltration vector [4][5][6]. Commercial systems have used ultrasonic beacons for device pairing and presence detection. Harmony inverts the threat model that this literature warns about: listening is explicitly opt-in and user-started, decoding is local, the payload is readable routing data rather than a covert identifier, and the consequence of a successful decode is a request that still requires human approval. The prior art is nonetheless a design constraint: it demonstrates both that the channel works and that deploying it without consent and transparency is unacceptable.

Agent triggering and workflow systems. Contemporary agent frameworks trigger execution through API calls, scheduled jobs, webhooks, or user prompts inside an application. Workflow and queue systems provide the routing, claim, retry, and audit semantics that agent execution increasingly inherits. Harmony does not replace any of this machinery. It adds a new trigger surface in front of it: the runtime in our prototype is an ordinary task-queue server with node registration, claims, claim expiry, cooldowns, and receipts. The novelty is confined to how a task request originates — from a cue carried by media — which is precisely the layer existing systems leave to applications.

3. Design Goals

The prototype was built against five goals.

G1. Decouple asset lifetime from task lifetime. A cue embedded once must be able to address work that did not exist when the media was authored. This rules out embedding tasks directly as the primary mechanism and motivates the workflow cue packet type.

G2. Keep the channel honest about its capacity. The near-ultrasonic FSK channel is deliberately low-rate. It is suited to compact routing packets, not bulk data, and the system design accepts that constraint rather than hiding it: payloads are routing fields, integrity data, and policy fields, and airtime is a first-class scheduling concern in the encoder.

G3. Local detection, minimal disclosure. Decoding happens on the listening device. The runtime learns that a validated cue was presented by a registered node; it does not receive audio.

G4. Approval before execution, always. No decoded signal executes anything by itself. The cue requests; the runtime assigns; the human approves; the executor runs.

G5. A record for every run. Each completed execution produces a receipt binding node, task, output, and timestamps, so that media-initiated work is auditable in the same way as console-initiated work.

4. The Signal Layer

4.1 Packet types and compilation

Authoring begins in Encoder Studio, where a rules-based compiler maps an instruction or intent into a structured HMY2 packet. The compiler is deterministic: it does not call a language model, and its content analysis uses keyword rules over title, tags, intention, and filename rather than semantic understanding of audio. Two packet types are defined.

Packet type	Carries	Best fit
Static task	Action and parameters, task identity, organization, agent suite, approval and integrity fields	A fixed instruction authored with the media

Packet type	Carries	Best fit
Workflow cue	Organization, workflow, channel, agent suite, a claim_next_task execution policy, and integrity fields	A reusable media cue addressing an evolving task queue

A workflow cue compiled from the instruction "wake the music release queue" takes the following form (routing-field excerpt; a complete packet additionally contains nonce, cooldown, time-to-live, hash, signature, and CRC fields):

```
{
  "protocol": "HMY2",
  "packet_type": "workflow_cue",
  "org_id": "demo_org",
  "workflow_id": "music_release_queue",
  "channel": "music",
  "agent_suite": "CREATOR",
  "execution_policy": "claim_next_task",
  "approval_required": true
}
```

4.2 Encoding

The encoder serializes the compact packet into bits and modulates them onto the media's audio samples using binary frequency-shift keying: one tone represents 0 and another represents 1.

Table 2. Default HMY2 channel parameters (prototype).

Parameter	Value
Tone for 0	17,400 Hz
Tone for 1	18,600 Hz
Symbol duration	80 ms
Packet amplitude setting	−42 dB
Symbol fade	5 ms
Framing	16-bit sync + 16-bit payload length + payload bits

At the default symbol duration the raw signaling rate is 12.5 bits per second. A 250-character compact payload therefore requires approximately 162.56 seconds of airtime including framing. Media duration, insertion timing, packet size, and repeat spacing must be planned together; the encoder's scheduler warns when a packet extends beyond the media, but such a warning is not a guarantee that a payload will fit or decode. These frequencies are near-ultrasonic: audibility depends on the listener, the playback equipment, the source content, and the settings, and universal inaudibility is not claimed. The payload is readable data, not encrypted content.

Two distinctions matter for interpreting the system. First, the FSK packet channel is separate from the encoder's sound-design layers (for example, tonal beds and binaural or isochronic modes offered by the audio production tools): only the FSK payload carries machine-readable routing fields, and frequency labels or resonance modes are not claimed to produce medical, cognitive, or behavioral effects. Second, for video, the signal is carried in the audio track only. The system implements no image-pixel encoding, no video-frame watermarking, and no signaling in silent media.

4.3 Export and inspection

Audio is mixed in the browser and exported as 16-bit WAV; MP3 is supported as an input, not as an implemented export path. For video, an FFmpeg WebAssembly pipeline extracts the audio track, applies the same encoding, and remuxes the result into an MP4, attempting stream copy with a re-encode fallback; actual format support depends on the browser and FFmpeg path.

The Signal Inspector recovers packets from media files directly, without requiring a sidecar manifest, and its reports distinguish packet validation results, workflow routing fields, and frequency-layer observations. Optional .harmony.json export metadata supports file-hash checks and packet summaries. Because inspection needs no manifest, any party holding the media can verify what cue it carries — a transparency property we consider essential for a trigger mechanism embedded in published content.

5. System Architecture and Implementation

The prototype comprises six components, all implemented in code and reviewed at source level in October 2026.

Component	Implemented capabilities
Encoder Studio	Browser media import, frequency-layer authoring, task compilation, sonic packet encoding, WAV export, video audio processing; waveform regions, presets, stereo/binaural/isochronic modes, vocal clips, ambience, preview and limiting; an experimental layered-synthesis Moment Resonance Engine
Signal Inspector	Packet decoding from media, validation and routing-field display, frequency-layer scanning, JSON/Markdown report export
Verification and manifests	Media and profile hashes, export metadata, packet summaries, compatibility reports, execution previews
Listening node	User-started microphone listening with local packet decoding (in Studio), and a Chrome/Edge Manifest V3 browser extension providing microphone and tab capture, local validation, manual claims, template execution, receipts, and proof exports
Reliability bench	Simulated gain, filtering, noise, resampling, codec-like and playback-path degradation with decode scoring
Runtime server	Organization queues, task creation, node registration, claims, claim expiry, cooldowns, receipts, and test points; seeded demo workflows, an operations dashboard, and container deployment configuration

The end-to-end flow is:

1. An author compiles a task or workflow cue and encodes it into media audio.
2. The encoded media is published or played through ordinary channels.
3. A user starts an opt-in listening node, which decodes and validates the packet locally.
4. For a workflow cue, the node presents the routing fields and validation flags to the runtime and requests the next task.
5. The runtime checks node credentials, validation flags, cooldowns, and queue eligibility, then assigns an approved task.
6. The user confirms local execution.
7. The executor produces output and the system records a receipt (with test points in the prototype).

Static task packets follow a separate receipt path rather than claiming a queued task. The code defines channel presets for a range of business and creative functions — marketing, creator, content, sales, support, finance, legal, operations, engineering, security, analytics, human resources, education, commerce, community, music, video, gaming, research, personal, and custom — which are routing presets and suite definitions, not independently integrated products.

6. Execution and the Runtime Contract

The prototype executor is template-based. Its local actions include research briefs, summaries, captions, campaign outlines, drafts, image-prompt text, guided scripts, and proof logging. These outputs are deterministic template results, not model inference; an image-prompt action produces text and does not generate an image. Compiler action names and executor capabilities are not completely identical, so a compiled action alone does not establish an end-to-end feature — a gap we record rather than paper over.

The executor's position in the architecture is deliberately narrow, so that it can be replaced without redesigning the system. A model-backed executor can be substituted behind the same claim, approve, execute, and receipt contract, adding model selection, scoped tool permissions, budgets, retries, and output validation. That substitution is planned work (Objective 5, Section 8), not a present capability, and no external model service,

GPU platform, publishing connector, or business-system integration is implemented in the reviewed code.

The runtime enforces the operational semantics of the queue: node registration with credentials, claim assignment with expiry, per-node and per-workflow cooldowns, and receipt deduplication. Persistence in the prototype is JSON file state rather than a production database, and container configuration exists although a successful live deployment was not verified in the source review.

7. Security and Privacy Analysis

A mechanism that lets published media request work on nearby devices must be analyzed as a security system, not only as a convenience. We state the prototype's mechanisms and its known weaknesses in the same place.

Integrity mechanisms present. Packets carry CRC16 checks, shortened SHA-256-derived packet hashes, and a proof-of-concept signature field. The executor applies action allow and block lists; live publishing, payments, wallet transfers, deployment, deletion, unapproved email, and legal execution are blocked in the prototype's action model. Execution requires manual confirmation. Node authentication uses a shared secret, and claims expire.

Known weaknesses. (i) The POC signature is a six-hex-character hash derived using a public constant salt; it is not a private-key digital signature and provides no production publisher authentication. (ii) The runtime accepts client-supplied packet-validation booleans; it does not independently decode audio or establish trustworthy packet provenance, so a malicious client can assert validity the server cannot check. (iii) Node authentication is a POC shared secret; production identity, tenant authorization, and hardened endpoint access require further work. (iv) Packet time-to-live is represented, but the compact format omits the original issue timestamp and parsing assigns the current time, so strong expiration and replay protection are not established by those fields alone. (v) Receipts are application records of client assertions, not independently verified proof of computation or of model correctness. (vi) Rewards in the prototype are test points; no payments, token payouts, or on-chain settlement are implemented.

Privacy posture. The privacy properties are architectural rather than policy-based: listening is opt-in and user-started; decoding is local; raw audio does not leave the device in the reviewed flows — the runtime receives metadata, task information, and receipts. The payload is readable by design, and the Signal Inspector lets any holder of the media inspect its cue, which removes the covertness that made ultrasonic beacons objectionable in prior work [4][5]. The residual risks are those of any trigger channel: spoofed cues (mitigated in production by the authentication work in Objective 4), unwanted triggering in shared acoustic spaces (mitigated by opt-in listening, cooldowns, and approval gating), and the low but nonzero audibility of near-ultrasonic tones for some listeners and equipment.

8. Evaluation Status and Validation Program

We report the evidence exactly as it stands, because the gap between a demonstrated loop and a validated channel is the paper's central engineering fact.

What has been demonstrated. The complete loop — cue encoded into audio, decoded by a listening node, used to claim an approved queue task, executed by the local executor, and recorded as a receipt — has been demonstrated end-to-end internally (June 2026). The repository additionally contains an encoded WAV, manifest, metadata, listener log, summary, and reliability report for a sample track ("Believe That," by Ka'Elum).

What the saved artifacts do not establish. The saved summary for that sample reports verification_compatible: true alongside a failed reliability grade, a 0% pass rate, zero listener events, and no agent execution. The reliability report records 0 of 15 scenarios passed, including clean WAV, and — decisively — expected_packet_count: 0, a test-configuration defect that limits what the run can be interpreted as measuring at all. These artifacts document a generated media package and reported compatibility; they do not establish a successful acoustic listener-to-agent run or reliable survival through streaming-platform processing. The bench's platform and codec scenarios are simulations: a scenario labeled with a platform's name is not evidence of an actual upload, transcode, and download test on that platform. No fresh end-to-end run was performed for the source review on which this paper is based.

Validation program. Converting the prototype into a measured channel requires five objectives, each with acceptance criteria:

1. **Instrumented baseline.** Repeatable clean-file encode and decode runs with correctly specified expected packet counts, reporting success rate, detection latency, and packet airtime across repeated trials, including false-detection controls on unencoded media.
2. **Real-world compatibility.** Measured speaker-to-microphone and tab-capture runs across commodity

phones, laptops, and speakers at documented distances and volumes, and actual upload, transcode, and download cycles on the streaming platforms media travels through — producing a compatibility matrix by codec, platform, device, and acoustic environment.

3. **Robust signaling.** Evaluation and implementation of error correction, repeated cue scheduling, and reduced airtime, quantifying each scheme's reliability and audibility trade-offs against the baselines from Objectives 1 and 2.
4. **Production authentication and verification.** Authenticated publishers with full-strength signatures, server-side packet verification, durable replay protection, and tenant-scoped authorization, red-teamed against spoofing, replay, and false-receipt attacks.
5. **Model-backed execution pilot.** A real model adapter behind the existing claim, approve, and receipt contract, with scoped permissions, budgets, and output validation, run as one narrow end-to-end pilot — a media cue addressing a creator workflow queue, producing a model-generated deliverable, approved by the user, recorded in a receipt, with success rate and latency measured through a real playback path.

9. Discussion

Media as a trigger surface. The useful analogy for Harmony is workflow infrastructure, not a media feature. The runtime is the execution layer — queues, task assignment, approved execution, and verified records. The cue is the routing layer: one embedded signal routes a playback device to an organization, a workflow, a channel, and an agent suite, and what it routes to can change without changing the media. Under that framing, a media catalog is not only content; it is a very large set of pre-distributed, durable entry points into workflow queues, hosted on playback infrastructure that already exists at global scale. A song released today can address a release-promotion queue this month and a different queue next year; the asset is authored once, and the addressing is stable while the work evolves.

Why the low rate is acceptable. The channel's roughly 12.5 bits per second would be disqualifying for data transfer and is irrelevant for routing. A workflow cue needs only to identify itself and its policy; everything voluminous — the task definition, its inputs, its outputs — travels over ordinary network connections after the cue has done its one job. Evaluating media-carried signals as a data channel misses the design point; they are an addressing channel.

Consent as a differentiator. The ultrasonic literature is dominated by covert deployments, and the resulting public and regulatory reaction has been hostile for good reason. Harmony's design treats the properties that made beacons objectionable — silent listening, covert identifiers, undisclosed data collection — as defects to be engineered out: opt-in listening, local decoding, readable payloads, public inspectability of cues, and human approval before any execution. Whether that posture is sufficient in shared spaces and at platform scale is an empirical and policy question the validation program must engage, not a claim this paper settles.

10. Limitations

This is a prototype report, and its limitations are substantial. The executor performs template actions, not model inference. Authentication is proof-of-concept grade, and the runtime trusts client-reported validation. Receipts record client assertions rather than verified computation. Channel reliability is unmeasured under real acoustic and platform conditions, and the one archived reliability run is not interpretable as a pass or fail rate because of its configuration defect. Compatibility results in the bench are simulations. Audibility of the near-ultrasonic channel is listener and equipment dependent and has not been formally evaluated. Video support carries signals in the audio track only, and MP3 export is not implemented. Finally, the system's author is also its inventor and the founder of the company developing it; the evidence discipline of Section 8 is the paper's safeguard against the optimism that conflict invites, and independent replication is the appropriate next check.

11. Conclusion

Harmony demonstrates that a media file can carry a small, inspectable, routing-only cue that connects playback to approved agent work through an ordinary queue runtime — and that the interesting engineering is everything around that fact: keeping the cue stable while work changes, keeping listening consensual and local, keeping execution approval-gated and recorded, and measuring honestly whether the acoustic channel survives the real world. The prototype stack implementing the full loop exists and has been run end-to-end internally. What remains is validation: an instrumented baseline, real-world compatibility measurement, robust signaling, production authentication, and a model-backed execution pilot. If that program succeeds, media stops being a passive endpoint of distribution and becomes addressing infrastructure for agent systems — every speaker and screen a potential, consenting entry point into work.

Declarations

Patent. The system described is the subject of USPTO Provisional Application No. 63/587,817 (Harmony), converted to a nonprovisional application; patent pending. The author is the inventor and holds the application personally.

Competing interests and research home. This is the author's private research, conducted independently. The author is the inventor of Harmony and holds its patent application personally; no company sponsors, funds, or houses this research program, and no company affiliation is claimed for it. The author is separately the founder of companies in related areas, which are not parties to this work. The university affiliation reflects the author's enrollment in its M.S. in Computer Science program; this work was conducted independently of the university as well.

Data and code availability. System description and evidence boundaries are drawn from the project's source repository and its October 27, 2026 source review. Public release of code and test artifacts is at the author's discretion pending the validation program in Section 8.

References

[1] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, *Digital Watermarking and Steganography*, 2nd ed. Burlington, MA: Morgan Kaufmann, 2008.

[2] M. Arnold, S. Wolthusen, and M. Schmucker, "Techniques and applications for digital audio watermarking," (survey literature on audio watermarking techniques, capacity, and robustness trade-offs).

[3] N. Cvejic and T. Seppänen, "Digital audio watermarking in the copyright protection era," *EURASIP Journal on Information Security*, 2007.

[4] D. Arp, E. Quiring, C. Wressnegger, and K. Rieck, "Privacy threats through ultrasonic side channels on mobile devices," in *Proc. IEEE European Symposium on Security and Privacy (EuroS&P)*, 2017.

[5] M. Hanspach and M. Goetz, "On covert acoustical mesh networks in air," *Journal of Communications*, vol. 9, no. 10, 2014.

[6] L. Deshotels, "Inaudible sound as an attack vector" (on near-ultrasonic sound as an attack and command channel), 2014.

[7] M. Cheeks, "Harmony / Solfeggio AI: system source review and repository documentation," personal research archive, technical review, October 27, 2026. (Source basis for implementation and evidence statements in Sections 5–8.)

[8] M. Cheeks, "Harmony: Media-Carried Workflow Signals for Distributed AI Agent Execution," project pitch (unpublished draft v3), October 27, 2026. (Source basis for the validation program structure in Section 8.)