OpenSound Protocol

A Decentralized, Open Standard for Music Distribution

Public Draft — Version 0.1

February 2026

Authors

Conceived in conversation between a human author and Claude (Anthropic AI, claude.ai) — February 26, 2026

AI CONTRIBUTION This whitepaper was co-authored with Claude, an AI assistant developed by Anthropic. The protocol concepts, architecture decisions, and problem framing emerged through a live dialogue session. Claude contributed the technical decomposition, protocol layer design, governance framework, and the composition-first philosophy. This document is released as a public draft and is intended as a starting point for community discussion — not a finished specification.

Abstract

The music industry has long suffered from a structural imbalance: artists create, platforms control, and intermediaries extract value at every layer. Centralized streaming platforms, while convenient, impose opaque royalty structures, algorithmic gatekeeping, and single points of failure that can disappear any artist or any catalog at will.

OpenSound Protocol (OSP) proposes a different foundation — one where music distribution is a protocol, not a product. Inspired by the architecture of HTTP, BitTorrent, and ActivityPub, OSP defines a minimal, composable standard for music identity, content distribution, metadata, and discovery. No company owns it. No server controls it. Any client, any node, any fork is a first-class citizen.

This document is a first draft and an open invitation. Read it, argue with it, improve it.

1. The Problem

1.1 Centralization Tax

Every dominant music platform today is a centralized silo. Spotify, Apple Music, Tidal, and their peers control catalog access, royalty rates, algorithmic promotion, and artist relationships. Artists receive between $0.003 and $0.005 per stream on average. The platform captures the

rest.

This is not a business model complaint — it is a structural observation. When distribution requires permission from a platform, the platform becomes a tollbooth on culture itself.

1.2 Why Previous Alternatives Failed

Earlier attempts at decentralized music distribution failed for identifiable reasons:

- Napster, LimeWire: centralized components (index servers, company) made them attackable
- Early BitTorrent music: no streaming capability, poor UX, no metadata standard
- Audius: decentralized in name but relies on a token economy controlled by insiders
- Blockchain music projects: prioritized financialization over actual music distribution

The pattern is consistent: projects tried to build a platform or a business when they should have been building a protocol.

1.3 The Protocol Gap

The internet has open protocols for nearly every form of communication: email (SMTP), web (HTTP), video streaming (HLS, WebRTC), social networking (ActivityPub), identity (DID). Music distribution has no equivalent. This is the gap OSP intends to fill.

2. Design Philosophy

2.1 Protocol, Never Platform

OSP's primary commitment is to remain a protocol specification, not a product. This means: no company owns it, no foundation controls the network, no token gates access. The specification is public domain. Any implementation is valid. Any fork is legitimate.

This is the HTTP model. Nobody can shut down the web by suing W3C. Nobody can take down email by targeting IETF. OSP aims for the same structural resilience.

2.2 Compose, Don't Invent

Every component of OSP should, where possible, delegate to an existing proven standard. We do not need a new identity system — DIDs exist. We do not need new content addressing — IPFS exists. We do not need new P2P streaming — WebTorrent exists. OSP's job is to define how these pieces fit together for music specifically.

2.3 Minimize the Core

The core protocol should be as small as possible. Only what is necessary for interoperability must be standardized. Everything else — recommendations, discovery, payments, social features — belongs in optional extension layers that implementors may or may not adopt.

2.4 Legal Architecture by Design

OSP is designed so that the protocol itself is neutral software. It defines data formats and communication patterns. What content flows through any given implementation is the responsibility of that implementation's operator and users — precisely as with BitTorrent, HTTP, or SMTP.

> **LEGAL NOTE** OSP does not host, index, or distribute content. It defines how content can be hosted, indexed, and distributed by others. The legal precedent of Betamax, Grokster v. MGM, and the treatment of BitTorrent as neutral software all support the viability of this architecture.

3. Protocol Architecture

3.1 Layer Model

OSP is organized in four layers, each building on the one below:

```
Layer 3 — Applications     (clients, players, apps, stores)
Layer 2 — Social           (playlists, follows, comments, discovery)
Layer 1 — Distribution     (P2P streaming, seeding, CDN fallback)
Layer 0 — Identity/Content (DIDs, IPFS hashes, cryptographic signatures)
```

Only Layer 0 is mandatory for OSP compliance. Higher layers are optional and extensible. An implementation that only implements Layer 0 can still interoperate with all other OSP nodes.

3.2 Layer 0 — Identity and Content Addressing

Artist Identity

Artists are identified by W3C Decentralized Identifiers (DIDs). A DID is a globally unique, self-sovereign identifier that requires no central registry. Artists generate their own DID and control

the associated key pair.

```
did:key:z6MkhaXgBZDvotDkL5257faiztiGiC2QtKLGpbnnEGta2doK
```

Content Addressing

All audio content is addressed by its IPFS Content Identifier (CID) — a cryptographic hash of the file's contents. This means content is immutable, self-verifying, and location-independent. The same file has the same CID everywhere, forever.

Track Record

A Track Record is the core data structure of OSP. It is a JSON-LD document signed by the artist:

```
{
  "type": "osp:Track",
  "version": "0.1",
  "id": "osp:track:Qm...",
  "title": "Song Name",
  "artist": "did:key:z6Mk...",
  "duration": 243,
  "format": "audio/opus",
  "cid": "QmXyz...ipfs-content-hash",
  "torrent": "magnet:?xt=urn:btih:...",
  "released": "2026-01-15",
  "license": "CC-BY-SA-4.0",
  "signature": "base64-artist-signature"
}
```

The artist's cryptographic signature over the Track Record means: no platform can forge a release on behalf of an artist, and no platform can suppress a release the artist has published.

3.3 Layer 1 — Distribution

Content distribution in OSP is peer-assisted by default. The primary mechanism is WebTorrent, which runs BitTorrent over WebRTC and works in any modern browser without plugin installation. Nodes that play a track automatically contribute bandwidth to seed it — as Spotify quietly did with its P2P cache for years.

The distribution stack:

- Primary: WebTorrent P2P between active listeners
- Secondary: IPFS content retrieval from the broader IPFS network
- Fallback: HTTP CDN for cold/unpopular content (operator-configurable)

Cold content — tracks with few listeners — is the hard problem. OSP addresses this through community seed nodes, optional incentivized seeding extensions, and CDN fallback. No single solution is mandated at the protocol level.

3.4 Layer 2 — Metadata and Discovery

Discovery in OSP is built on a distributed hash table (DHT), the same mechanism used by BitTorrent's decentralized tracker. Nodes announce the Track Records they hold, and any node can query the DHT to find tracks matching a given artist DID, title, or genre tag.

This layer is intentionally minimal. Search quality, recommendations, and curation are application-layer concerns and are explicitly out of scope for the core protocol. Different clients will build different discovery experiences on top of the same underlying DHT.

3.5 Layer 3 — Applications

Any application that speaks OSP is a valid OSP client. This includes web apps, mobile players, CLI tools, hardware devices, or server-side indexers. The protocol places no constraints on the application layer beyond correct implementation of the data formats.

4. Governance

4.1 The Protocol Foundation

OSP should be stewarded by a lightweight nonprofit foundation whose mandate is strictly limited to:

- Maintaining the protocol specification
- Running a public RFC process for proposed changes

- Holding the OSP trademark
- Nothing else

The foundation does not operate nodes, does not host content, does not run a marketplace, and does not have the ability to approve or deny implementations. Compliance with the spec is self-certifying.

4.2 Change Process

Protocol changes follow a public RFC model inspired by Nostr's NIPs and Bitcoin's BIPs:

- Any person may submit an OSP Improvement Proposal (OIP)
- OIPs are discussed publicly in the open repository
- Changes to the core spec (Layer 0) require supermajority consensus among active maintainers
- Extension layers (Layers 1-3) are more permissive and can be adopted optionally

4.3 Licensing

The protocol specification is released under CC0 (public domain). The reference implementation is released under the MIT license. There are no patent claims. No contributor agreement transfers rights to the foundation.

5. Reference Implementation

5.1 Scope

The reference implementation (osp-node) demonstrates the protocol end-to-end. It is not intended to be production software — it is the canonical proof that the spec works. Other implementations are primary citizens; osp-node is just the first.

5.2 Proposed Stack

- Runtime: Node.js (initial) with Rust rewrite planned for performance-critical paths
- P2P streaming: WebTorrent.js
- Content addressing: Helia (lightweight IPFS in JavaScript)
- DHT: libp2p Kademlia
- Identity: @digitalbazaar/did-io for DID generation and resolution
- Web client: standard browser APIs, Web Audio API for playback

5.3 Build Order

The reference implementation will be built in the following sequence:

- Phase 1: CLI publish and resolve — prove Track Records work end to end
- Phase 2: CLI playback — prove streaming over WebTorrent is viable for audio
- Phase 3: Minimal web client — prove the stack works in a browser
- Phase 4: DHT discovery — prove decentralized search is functional
- Phase 5: Documentation and developer onboarding

6. Open Questions

This draft intentionally leaves several questions open for community discussion:

6.1 Content and Licensing

OSP is content-neutral. The reference implementation and official documentation will focus on Creative Commons and artist self-published content. The question of how implementations handle commercially licensed music is left to those implementations and their legal counsel.

6.2 Cold Content Seeding

Who seeds unpopular tracks when there are no active listeners? Options include: volunteer community seed nodes, opt-in incentivized seeding, artist-run origin nodes, or CDN fallback. The core protocol does not mandate a solution — extension proposals welcome.

6.3 Spam and Metadata Quality

A fully open metadata layer will attract spam. How do clients filter signal from noise? Options include web-of-trust reputation systems, social graph signals, or curated index services. This is an application-layer problem, but one worth solving early.

6.4 Payments

OSP does not define a payment layer. An optional extension using Lightning Network micropayments or donation mechanics is under consideration but is explicitly out of scope for v0.1.

7. Conclusion

The music industry does not need another platform. It needs a protocol — one that is as durable as HTTP, as unstoppable as BitTorrent, and as open as email.

OpenSound Protocol is a proposal, not a product. The goal is not to build a company or capture a market. The goal is to write a spec that is so clear and so useful that independent developers build a dozen different clients on top of it — and none of them can be shut down by suing the spec.

The hardest problems here are not technical. They are social: bootstrapping content, building community, and creating a governance model that survives success. These require people, not code.

If this document resonates with you — if you are an artist, a developer, a researcher, or someone who simply believes music should be free from centralized control — this is your invitation to participate.

The repository is open. The spec is public domain. Come build.

Acknowledgements

This whitepaper was initiated in a live conversation on claude.ai on February 26, 2026. The human author brought the question: why did torrent streaming audio never work, and could we build it differently? Claude (Anthropic's AI assistant, claude-sonnet model) contributed the analysis of prior failures, the composition-first design philosophy, the four-layer protocol architecture, the governance framework modeled on Nostr and Bitcoin, and the drafting of this document.

The ideas here draw on the work of the BitTorrent, IPFS, WebTorrent, Nostr, ActivityPub, and W3C DID communities. None of this is invented — it is assembled, with gratitude.

References

- W3C Decentralized Identifiers (DIDs) v1.0 — https://www.w3.org/TR/did-core/
- IPFS Whitepaper — Juan Benet, 2014
- WebTorrent — https://webtorrent.io
- ActivityPub — W3C Recommendation, 2018

- Nostr Protocol — https://nostr.com
- libp2p — https://libp2p.io
- Bitcoin BIP Process — https://github.com/bitcoin/bips
- Helia (IPFS in JS) — https://github.com/ipfs/helia