API REFERENCE

ERICADE Radio API v2 Documentation

Base URL: `https://api.ericade.net/` — all v2 endpoints live under `/radio/v2/`. See also the v1 documentation for the original `/radio/` endpoints.

Updated: 2026-09-19

Contents

Conventions

Unless stated otherwise, every v2 endpoint follows the same pattern:

- **Method:** `POST` with a JSON body.
- **Content type:** the response is served as `application/json; charset:UTF-8`.
- **CORS:** `Access-Control-Allow-Origin: *` is set on all endpoints — except the user account and content administration endpoints, which carry credentials and answer only to radio.ericade.net.
- **Routing:** the path after `/radio/v2/` is matched by prefix. An unrecognised path returns HTTP 400 with `“The endpoint you have requested does not exist.”`

> The user account endpoints (`authenticate`, `account`, `profile`, `privileges`) and the content administration endpoints (`podcast`, `songdata`, `artistdata`, `news/<action>`, `comment`, `message`) follow stricter rules of their own: one envelope, meaningful status codes, no caching. See how they differ.

Response envelopes

Endpoints that return data echo their JSON payload directly (for example `{"Tracks": [ ... ]}`). Endpoints that only perform an action use one of *three* different envelope shapes — check which one applies before writing a client:

SHAPE	USED BY	EXAMPLE
`message` / `subcode` / `submessage`	All failures on the data endpoints, plus nextup.	`{"message":"Failure","subcode":"FALSE","submessage": "You need to specify a StationID."}`
`Success` / `Message`	addrequest and settings. Note the capitalised keys.	`{"Success":"Success","Message":"Override updated."}`
`success` / `message`	rating. `success` is a real JSON boolean here, not a string.	`{"success":true,"message":"Stars updated."}`

Status codes

CODE	MEANING
`200`	Success. The body is the endpoint’s payload or a success envelope.
`204`	Nothing has changed since the `Sequence` value you sent. Keep displaying the previously received data.
`400`	Validation failure, missing/invalid credentials, unknown endpoint, or a caller outside the allowed IP range.
`429`	Rate limited — see below.

> **Do not branch on the status code alone.** nowplaying, search, songs, the track listing and the artist endpoints return their *failure* envelope with HTTP 200. Always check the `message` field of the body.

Rate limiting (HTTP 429)

Every endpoint runs a shared per-IP throttle before doing any work. Each caller gets a rolling window with a maximum request count; exceeding it starts a temporary ban, and the request that triggered the ban is itself rejected. While banned, every request — to any endpoint — returns:

`HTTP 429 Too Many Requests` + `Retry-After: <seconds>`

```
{
  "message": "Failure",
  "subcode": "Too Many Requests",
  "submessage": "You have been throttled due to too many requests. Please try again in 600 seconds."
}
```

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Retry-After` (header)	Seconds remaining on the ban. Honour this value rather than retrying on a fixed timer.	600
`message`	Always `Failure`.	Failure
`subcode`	Always `Too Many Requests`. Use this to distinguish a throttle from an ordinary failure.	Too Many Requests
`submessage`	Human-facing message including the remaining wait, suitable for display.	You have been throttled due to too many requests. Please try again in 600 seconds.

> Retrying while banned does not extend it, but it does not succeed either — back off until `Retry-After` elapses. Polling clients should use the `Sequence` mechanism on Now Playing, or the Station Data WebSocket, instead of a tight poll loop. Whitelisted addresses bypass the throttle entirely.

Station IDs

STATIONID	STATION
`1`	ericade.radio — 24/7 tracked music
`2`	Best of ericade.radio — podcast station

Authentication

All endpoints require a JWT token. Obtain one by making a GET request to the token endpoint:

`GET` `https://radio.ericade.net/get-token/`

The response is a raw JWT string. Pass it in your POST body — but note that **the field name is not the same on every endpoint**. Most read `Token`; a handful read `Password`. Sending the wrong one leaves the call unauthenticated.

FIELD NAME	ENDPOINTS
`Token`	nowplaying, getnextup, songs/<trackid>, search, songs, artists, artists/<artistid>, sitemapartists, songhistory, playout, new-songs, news/<id>, stats, stats/<filtertype>, listening, requests, rating
`Password`	news, addrequest, settings

The ingest endpoints do not use a JWT at all — they authenticate with a pre-shared secret, also sent as `Password`, and are additionally restricted by IP.

> Token replay protection is currently disabled, but reusing tokens across sessions is still inadvisable.

> This token identifies a *browser*. Logging in a *person* yields a second, different token — `UserToken` — described under User Accounts.

Now Playing

Returns what is currently playing on a station, with comprehensive metadata about the artist and track. Also supports a bandwidth-saving sequence mechanism that returns HTTP 204 when nothing has changed.

`POST` `https://api.ericade.net/radio/v2/nowplaying/`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	The station to get data from.	1 = ericade.radio 2 = Best of ericade.radio
`Sequence`	Yes	Bandwidth-saving mechanism. Send 0 to always receive a full response, or send the `TimeStamp` value from the previous response. If the current track's timestamp matches the sent value the API returns HTTP 204 (no body); if it differs, a full response is returned with an updated `TimeStamp`.	Set to 0 on non-metered connections.
`DisableBufferCompensation`	No	0 = use buffer compensation (default), 1 = disable. The API holds the current track for a preset number of seconds to compensate for stream buffering.	Should always be 0 or omitted unless explicitly needed.
`ReturnPlayList`	No	0 = do not return `PlayList` (default), 1 = return `PlayList`.	Also needed for NowPlayingOnPlaylist to be populated.

Fixed server-side on this endpoint. `NowPlaying`, `ReturnNextUp`, `ReturnArtistDescription`, `ReturnArtistLongDescription` and `ReturnProductionNotes` are all forced to 1 by the router, so the response always includes `NextUp`, the artist biography and the podcast production notes. Sending those fields has no effect — earlier revisions of this document listed `NowPlaying`, `ReturnNextUp` and `ReturnArtistDescription` as caller-controlled, which was incorrect. Use Track by ID if you need to switch the artist description off.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 1,
  "Sequence": 0,
  "ReturnPlayList": 1
}
```

HTTP 204: Returned when the `Sequence` value matches the current track's `TimeStamp`. The caller should display the previously received data unchanged.

Response

Returns a `{"Tracks": [ ... ]}` object. The array normally contains one entry for now-playing requests.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Artist`	Complete artist field including all collaborators.	Cube
`Title`	Track title.	My pixels are weapons
`TrackUpdatedAt`	When the track's metadata was last updated.	2026-06-12 09:41:02
`TrackArtists`	Array of all artists linked to this track (a track can have more than one).	Array[]
`TrackArtists ⇒ ArtistID`	Unique artist identifier.	4821
`TrackArtists ⇒ Artist`	Artist name.	Cube
`TrackArtists ⇒ ShortDescription`	Short artist bio.	(text)
`TrackArtists ⇒ LongDescription`	Full artist bio.	(text)
`TrackArtists ⇒ Image`	The artist's picture — the same address the artist endpoints give, or empty when the artist has none. Pictures uploaded to `/images/artists/` have sized copies next to them (`<id>—150x150`, `-300x300`, …) for showing them small.	(https address, or empty)
`TrackArtists ⇒ CompositeRating`	Artist rating from 0.0–5.0.	4.6
`TrackArtists ⇒ Votes`	Number of votes the artist has received.	12
`TrackArtists ⇒ TotalPlays`	Number of times this artist has been played.	6789
`TrackArtists ⇒ Demozoo`	Link to the artist's profile on demozoo.org.	
`TrackArtists ⇒ Wikipedia`	Link to the artist's Wikipedia entry.	
`TrackArtists ⇒ CSDB`	Link to the artist's C64 Scene Database profile.	
`TrackArtists ⇒ OtherUrl`	Link to the artist's homepage or other point of interest.	
`TrackArtists ⇒ ModArchive`	Link to the artist's ModArchive profile.	
`TrackArtists ⇒ Bandcamp`	Link to the artist's Bandcamp page.	
`TrackArtists ⇒ SoundCloud`	Link to the artist's SoundCloud profile.	
`TrackArtists ⇒ YouTube`	Link to the artist's YouTube channel.	
`TrackArtists ⇒ Pouet`	Link to the artist's Pouet profile.	
`TrackArtists ⇒ isBroadcastProhibited`	Whether broadcasting this artist's tracks is prohibited.	0 or 1
`TrackArtists ⇒ isDownloadProhibited`	Whether downloading this artist's tracks is prohibited.	0 or 1
`StationName`	Station name in plain text.	24/7 tracked music
`Slug`	URL-friendly slug for the track.	https://ericade.radio/#/song/14820/mindstorm-trackerartist
`StationID`	Station number.	1 = ericade.radio 2 = Best of ericade.radio

ATTRIBUTE	DESCRIPTION	EXAMPLE
`WebStreamingOffset`	Web streaming offset for the station in seconds. Used for webplayers.	12
`StreamingOffset`	Streaming offset for the station in seconds. Used for all other players.	10
`TrackCanBeRequested`	Whether this track can be requested right now.	0 or 1
`RequestVerdict`	Human-readable reason why the track cannot be requested. Echo this string to the user.	Song was played recently
`CreationDateHR`	When the track was first added to the station, human-readable.	2026-03-30 15:45:04
`CreationDate`	When the track was first added, Unix epoch string.	1774885504
`AddedDate`	*(Deprecated) Same as* `CreationDateHR`.	2026-03-30 15:45:04
`TrackCanBeStarred`	Whether this track can receive a star rating right now.	0 or 1
`PodcastURL`	[Podcast] URL to the podcast audio file.	https://radio.ericade.net/Flashback/97.mp3
`EpisodeNumber`	[Podcast] Episode number. Empty for non-podcast tracks.	97
`Image`	Image URL for this track.	https://radio.ericade.net/images/demoscene.png
`BroadcastDate`	[Podcast] Broadcast date of the podcast episode.	2022-07-16
`TitleID`	*(Deprecated) Not in use.*	13770
`TrackID`	Unique, stable track identifier. Use this in all requests that reference a specific track.	15717
`TrackerType`	Type/format of the track.	Modern remix (demo scene)
`Type`	Stated/inferred song type from PlayIT Live. Not actively used.	Song
`Tags`	(Deprecated) Tags associated with the track.	“”
`TagList`	List of tags associated with the track.	[]
`PlayLength`	Length between Cue In and Cue Out in seconds (decimal, “.” separator).	193.12099773243
`PlayLengthHR`	Play length expressed as HH:MM.	03:13
`Batch`	Batch number auto-parsed from the Album field. Format: `<StationShortName>—<MonMMMYYYY>—<nn>`.	TERN-mar2026-03
`Album`	Album tag from the audio file’s metadata.	OriginalName:!Cube - My Pixels Are Weapons.ogg Imported:2026-03-29 (TERN-mar2026-03).
`About`	Short description from the song or podcast episode.	
`ProductionNotes`	[Podcast] Production notes. Empty for non-podcast tracks.	
`PlayList`	[Podcast] Chapter list for the podcast episode. Empty for non-podcast tracks. Only returns when a specific TrackID is requested.	00:00 Introduction…
`Transcript`	[Podcast] Full transcript of the episode. Empty for non-podcast tracks.	

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Footer`	[Podcast] Footer text/show notes appended to the episode. Empty for non-podcast tracks.	
`Equipment`	[Podcast] Equipment used to record the episode. Empty for non-podcast tracks.	
`CompositeRating`	Track rating from 0.0–5.0. Higher is better.	4.3
`Votes`	Number of votes the track has received.	2
`ArtistCompositeRating`	Artist rating from 0.0–5.0.	4.9
`ArtistVotes`	Number of votes the artist has received.	0
`TimeStamp`	Track start time as Unix epoch. Required for progress bar calculations and the Sequence mechanism.	1774885504
`TimeStampHumanReadable`	Track start time as a human-readable date string.	2026-03-30 15:45:04
`TrackTotalPlays`	Number of times this track has been played on the station.	313
`ArtistTotalPlays`	Number of times the artist has been played. Currently reflects the first artist only; will be revised.	206
`ArtistLongDescription`	Full artist biography. Only populated when `ReturnArtistDescription=1`. Currently reflects the first artist only; will be revised.	(text)
`ArtistShortDescription`	*(Deprecated) Do not use.*	
`Demozoo`, `Wikipedia`, `CSDB`, `OtherUrl`, `ModArchive`, `Bandcamp`, `SoundCloud`, `YouTube`, `Pouet`	Top-level copies of the *first* linked artist's external links. Prefer the per-artist values inside `TrackArtists`, which are correct for every collaborator.	https://demozoo.org/sceners/7637/
`ModArchiveTrueFileName`	The module's real filename as recorded on ModArchive.org. The enrichment job matches a track by the original filename recovered from the album string, which is not always the name ModArchive files the module under — this is the name ModArchive itself reports for the matched module. Empty when the track is not a tracker module. When populated, the data enrichment will use that name instead of the one inferred from the Album column when calling ModArchive.	virgill-bratgrumbeere.mod
`ModArchiveEnrichment`	Module metadata for this *track* pulled from ModArchive.org by a periodic background job. Distinct from the artist-level `ModArchive` link above — this is per-track data read out of the module file itself. Only meaningful for tracker-module tracks (an `original` file in `Media`); always present as an array of exactly one object, with empty/zero values before the track has ever been matched.	Array[]
`ModArchiveEnrichment ⇒ MA_Format`	Module format as reported by ModArchive.	Impulse Tracker
`ModArchiveEnrichment ⇒ MA_URL`	Direct URL to the module's page on ModArchive.	https://modarchive.org/index.php?request=view_by_moduleid&query=181206

ATTRIBUTE	DESCRIPTION	EXAMPLE
`ModArchiveEnrichment ⇒ MA_Date`	Date the module was added to ModArchive, as reported by ModArchive.	2005-08-14
`ModArchiveEnrichment ⇒ MA_Hash`	File hash ModArchive uses to identify the module.	3a1f9c2e…
`ModArchiveEnrichment ⇒ MA_Size`	File size as reported by ModArchive.	187 KB
`ModArchiveEnrichment ⇒ MA_SongTitle`	Song title as stored in the module's own metadata on ModArchive. May differ from the station's `Title`.	my_pixels_are_weapons
`ModArchiveEnrichment ⇒ MA_Instruments`	Instrument/sample list read out of the module file.	(text)
`ModArchiveEnrichment ⇒ MA_GenreID`	ModArchive's numeric genre identifier for the module.	3
`ModArchiveEnrichment ⇒ MA_Rating`	Average comment rating on ModArchive, rounded to the nearest whole number.	8
`ModArchiveEnrichment ⇒ MA_Comments`	Comment text pulled from ModArchive.	(text)
`ModArchiveEnrichment ⇒ MA_ReviewRating`	Review rating from ModArchive's reviews, rounded to the nearest whole number.	9
`ModArchiveEnrichment ⇒ MA_ReviewTotal`	Number of reviews the module has on ModArchive.	2
`ModArchiveEnrichment ⇒ MA_ImageUrl`	URL to an image associated with the module on ModArchive, when one exists.	
`ModArchiveEnrichment ⇒ MA_Artists`	Artist name(s) credited on ModArchive, comma-separated.	Cube
`ModArchiveEnrichment ⇒ MA_GuessedArtists`	ModArchive's best-guess artist name(s), used when the module could not be confidently attributed.	
`ModArchiveEnrichment ⇒ MA_LastUpdated`	Unix epoch of when this row was last written by the enrichment job.	1774885504
`ModArchiveEnrichment ⇒ MA_HasFile`	1 if ModArchive has a downloadable copy of the module, 0 otherwise. Not the same as having a local copy — see the `original` entry in `Media` for that.	0 or 1
`ModArchiveEnrichment ⇒ MA_Bytes`	File size in bytes, as reported by ModArchive.	191488
`ModArchiveEnrichment ⇒ MA_TimeStamp`	Unix epoch timestamp reported by ModArchive for the module.	1123977600
`ModArchiveEnrichment ⇒ MA_Found`	1 if the most recent enrichment lookup matched this track to a module on ModArchive, 0 if it did not. When 0, the other `MA_*` fields keep whatever ModArchive last told us, if anything.	0 or 1
`ModArchiveEnrichment ⇒ MA_LastSync`	When the enrichment job last looked this track up on ModArchive.	2026-06-01 03:12:47
`ModArchiveEnrichment ⇒ MA_NextUpdate`	When the enrichment job will look this track up again. Lookups are spread randomly between two weeks and three months apart so the whole library isn't re-swept at once.	2026-08-02 03:12:47

ATTRIBUTE	DESCRIPTION	EXAMPLE
`NowPlayingOnPlaylist`	[Podcast] [Best of ericade.radio(StationID=2)] returns the current chapter. Requires ReturnPlayList to be set to 1 in the call to work.	Love connection by Lavizh
`isListenerRequested`	"0" = regular rotation, "1" = listener-requested track.	"0" or "1"
`isOverride`	If set to 0 (default), the returned data will show the currently playing track. This is the normal behavior. If set to 1, it will show a preset text from the database. This is used to show special messages or tell the listener what live-show they're listening to.	"0" or "1"
`isLive`	This value only counts if the isOverride is set to 1. It will indicate that you are listening to a live show.	"0" or "1"
`isRemote`	This value only counts if the isOverride is set to 1. It will indicate that you are listening to a live show, broadcast from a remote location.	"0" or "1"
`isNew`	"0" = not new, "1" = new track.	"0" or "1"
`NewDays`	Shows the number of days a track is considered new.	7
`Podcast`	Present in all responses. Empty string for non-podcast tracks.	""
`musicURL`	Direct URL to the track's audio file.	https://radio.ericade.net/mods/cube-my_pixels_are_weapons.ogg
`musicURLFlac`	Direct URL to the track's FLAC audio file, when available. Empty if no FLAC version exists.	
`musicURLOriginal`	*(Not yet implemented) Intended to be the direct URL to the track's original tracker file (.mod, .it, .xm and so on). The resolver behind it is a stub, so it is currently an empty string on every endpoint — as is the* `original` *entry in* `Media`*. Do not build a download option on it yet.*	
`Media`	The three URLs above expressed as a typed list, in the fixed order mp3, flac, original. Convenience wrapper for players that iterate over available formats instead of reading each field by name.	Array[]
`Media ⇒ type`	Which representation this entry describes. The `original` entry's URL is currently always empty (see `musicURLOriginal`).	mp3, flac or original
`Media ⇒ url`	Direct URL to the file, or an empty string when the track has no file of that type.	https://radio.ericade.net/mods/custom.rmx
`isBroadcastProhibited`	Whether broadcasting this track is prohibited.	0 or 1
`isDownloadProhibited`	Whether downloading this track is prohibited. If 1, the music URLs and every `Media` entry are empty.	0 or 1

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Integrity`	Checksums of the track's audio file, in lowercase hex. Always present with all four keys. For a podcast episode the file is the MP3 at `PodcastURL`; for a song it is the MP3 at `musicURL`, worked out by the song scan. Each value is an empty string for a news post, for a track without an MP3 of its own, and for a track that has not been checksummed yet. See News & Podcasts for the checksum files published next to each episode.	Object{}
`Integrity ⇒ md5`	MD5 checksum, 32 characters.	ab8effe6ef0c5eb07803dd92ac14ac09
`Integrity ⇒ sha1`	SHA-1 checksum, 40 characters.	803d89f5701d0882a97e43ebfb231928e7ca4522
`Integrity ⇒ sha256`	SHA-256 checksum, 64 characters.	b886baff39c8ed3489d9275aa1cbd24af9b1a1885b02a0789d69242af7ff4176
`Integrity ⇒ sha384`	SHA-384 checksum, 96 characters.	f565bafcadb72fd55ef9948f1d63f2bf1ad060d02b40a63ca03209d125ec1abd730bedd0ee7c8947709953d029832160
`FileData`	What the station measured in the track's audio file, and the address of its waveform picture. **Always present with all twenty-one keys.** A podcast episode is measured when an administrator presses Scan podcast, a song by the song scan — a background job that works through the catalogue on its own. For a news post, and for a track that has not been measured (yet), `Status` is `none` and the rest is empty. It is part of a track wherever the API hands one out: here, in Track by ID, Search and List Tracks, in News & Podcasts and News by ID, in the songs inside the artist and statistics answers, and in the track events of the station-data WebSocket. A level is `null` when it was not measured — 0 dB is a level like any other, so it cannot stand for "unknown". **Every text in it is HTML-escaped**, the ID3 tags above all: they are whatever somebody typed into an audio file.	Object{}
`FileData ⇒ Status`	`none` (never scanned), `ok` or `failed`. Only `Timestamp` and `Message` are filled for a failed scan: numbers from an earlier, successful one are not handed out.	ok
`FileData ⇒ Timestamp`	When the file was scanned, Unix time. 0 = never.	1789745361
`FileData ⇒ Message`	Why the scan failed — a finished sentence, without server details. Empty otherwise.	
`FileData ⇒ FileSize`, `FileModified`	The size in bytes and the modification time (Unix) of the file that was measured.	57829120
`FileData ⇒ Duration`	Seconds, with decimals.	3614.302
`FileData ⇒ Bitrate`, `SampleRate`, `Channels`	Bits per second, Hz, and 1 (mono) or 2 (stereo).	128000
`FileData ⇒ LUFS`, `LRA`, `TruePeak`	EBU R128: integrated loudness (LUFS), loudness range (LU) and true peak (dBTP).	-16.4

ATTRIBUTE	DESCRIPTION	EXAMPLE
`FileData ⇒ LeftRMS`, `RightRMS`	Level of each channel over the whole programme, dBFS. `RightRMS` is `null` for mono.	-19.1
`FileData ⇒ ChannelDiff`, `StereoSeparation`	Left minus right, and side minus mid, in dB — explained under Scan podcast. `null` for mono.	0.6
`FileData ⇒ DR`	Dynamic range, dB: how far the peaks stand above the loud parts of the recording — the number of the Dynamic Range Meter ("DR12" is this value, rounded). Under 8 = heavily compressed, 14 and up = very dynamic; it does not depend on how loud the file is. How it is measured. `null` when it was not: the track was scanned before 2026-09-19 and not since, or it cannot be measured (silence, a recording of a few seconds). 0 is a value (a steady tone).	11.6
`FileData ⇒ WaveformURL`	The waveform as a chart — level in dB against time in hh:mm:ss — an opaque WebP of 1800×604 pixels (1800×364 for mono). Its letters are part of the picture: show it 760 pixels wide or wider, and let it scroll sideways on a narrow screen. `?v=` is the time of the scan, so the address changes when the picture does. An episode's picture is `<number>.webp`, a song's `<Guid>.webp`. Empty when there is none.	https://radio.ericade.net/images/waveforms/167.webp?v=1789745361
`FileData ⇒ ID3`	The file's tags, name => text: at most 40, names in lowercase, values cut at 500 characters and **HTML-escaped**. An empty object when there are none.	Object{}
`FileData ⇒ CoverArt`, `Codec`	1 when a picture is embedded in the file; the audio codec.	mp3
`NextUp`	Array of the next track's info. Only populated when `ReturnNextUp=1`. Station IDs and adverts are excluded.	Array[]
`NextUp ⇒ Artist`	Artist name of the next track.	MA2E
`NextUp ⇒ Title`	Title of the next track.	Thrilled 4 noise
`NextUp ⇒ AddedDate`	When the next track was first added, human-readable.	2022-04-09 14:41:38
`NextUp ⇒ CompositeRating`	Rating 0.0–5.0.	3.5
`NextUp ⇒ Voters`	Vote count.	0
`NextUp ⇒ Image`	Image URL.	https://radio.ericade.net/wp-content/uploads/2020/12/amiga_flashback-1024x1024.jpg
`NextUp ⇒ StationID`	Station the next track belongs to.	1
`NextUp ⇒ TrackerType`	Track type/format.	Amiga 4-channel module
`NextUp ⇒ TrackID`	Unique track identifier.	11971
`Playlog`	Array of recent play timestamps for this track on this station.	Array[]
`Playlog ⇒ timestamphr`	Human-readable datetime the track was last played.	2026-03-30 16:00:06

Example response


```
{
  "Tracks": [
    {
      "Artist": "Chris Huelsbeck",
      "Title": "Dressed to chill",
      "TrackUpdatedAt": "2026-06-12 09:41:02",
      "TrackArtists": [
        {
          "ArtistID": 4821,
          "Artist": "Chris Huelsbeck",
          "ShortDescription": "A true legend on the Amiga and other systems.",
          "LongDescription": "A true legend on the Amiga and other systems. He's most famous for the music
from Turrican I, II and III. His music can be found in many games and demos on various systems.",
          "CompositeRating": 4.6,
          "Votes": 12,
          "TotalPlays": 6789,
          "Demozoo": "https://demozoo.org/sceners/7637/",
          "Wikipedia": "",
          "CSDB": "",
          "OtherUrl": "",
          "ModArchive": "",
          "Bandcamp": "https://chrishuelsbeck.bandcamp.com/",
          "SoundCloud": "",
          "YouTube": "",
          "Pouet": "",
          "isBroadcastProhibited": 0,
          "isDownloadProhibited": 0
        }
      ],
      "StationID": 1,
      "StationName": "24/7 tracked music",
      "Slug": "https://radio.ericade.net/#/song/14051/dressed to chill-chris-huelsbeck",
      "WebStreamingOffset": 12,
      "StreamingOffset": 10,
      "TrackCanBeRequested": 0,
      "RequestVerdict": "Artist was played recently",
      "CreationDateHR": "2022-11-01 16:19:08",
      "CreationDate": "1667315948",
      "AddedDate": "2022-11-01 16:19:08",
      "TrackCanBeStarred": 1,
      "PodcastURL": "",
      "EpisodeNumber": "",
      "Image": "https://radio.ericade.net/images/rmx.png",
      "BroadcastDate": "",
      "TitleID": 12098,
      "TrackID": 14051,
      "TrackerType": "Modern remix",
      "Type": "Song",
      "Tags": "",
      "TagList": [
                  "Revision",
                  "Revision 2026"
                 ],
      "PlayLength": 273.39934240363,
      "PlayLengthHR": "04:33",
      "Batch": "TERN-oct2022-05",
```

```
      "Album": "OriginalName:custom.rmx Imported:2022-11-01 (TERN-oct2022-05).",
      "About": "",
      "ProductionNotes": "",
      "PlayList": "",
      "Transcript": "",
      "Footer": "",
      "Equipment": "",
      "CompositeRating": 0,
      "Votes": 0,
      "ArtistCompositeRating": 4.6,
      "ArtistVotes": 12,
      "TimeStamp": 1778939868,
      "TimeStampHumanReadable": "2026-05-16 13:57:48",
      "TrackTotalPlays": 216,
      "ArtistLongDescription": "A true legend on the Amiga and other systems. He's most famous for the
music from Turrican I, II and III. His music can be found in many games and demos on various systems.",
      "ArtistShortDescription": "A true legend on the Amiga and other systems. He's most famous for the
music from Turrican I, II and III. His music can be found in many games and demos on various systems.",
      "NowPlayingOnPlaylist": "",
      "Demozoo": "https://demozoo.org/sceners/7637/",
      "Wikipedia": "",
      "CSDB": "",
      "OtherUrl": "",
      "ModArchive": "",
      "Bandcamp": "https://chrishuelsbeck.bandcamp.com/",
      "SoundCloud": "",
      "YouTube": "",
      "Pouet": "",
      "ModArchiveTrueFileName": "virgill-bratgrumbeere.mod",
      "ModArchiveEnrichment": [
        {
          "MA_Format": "",
          "MA_URL": "",
          "MA_Date": "",
          "MA_Hash": "",
          "MA_Size": "",
          "MA_SongTitle": "",
          "MA_Instruments": "",
          "MA_GenreID": 0,
          "MA_Rating": 0,
          "MA_Comments": "",
          "MA_ReviewRating": 0,
          "MA_ReviewTotal": 0,
          "MA_ImageUrl": "",
          "MA_Artists": "",
          "MA_GuessedArtists": "",
          "MA_LastUpdated": 0,
          "MA_HasFile": 0,
          "MA_Bytes": 0,
          "MA_TimeStamp": 0,
          "MA_Found": 0,
          "MA_LastSync": "",
          "MA_NextUpdate": ""
        }
      ],
      "isListenerRequested": "0",
      "isOverride": 0,
      "isLive": 0,
```

```
      "isRemote": 0,
      "isNew": 0,
      "NewDays": 7,
      "Podcast": "",
      "musicURL": "https://radio.ericade.net/mods/custom.rmx",
      "musicURLFlac": "",
      "musicURLOriginal": "",
      "Media": [
        { "type": "mp3", "url": "https://radio.ericade.net/mods/custom.rmx" },
        { "type": "flac", "url": "" },
        { "type": "original", "url": "" }
      ],
      "isBroadcastProhibited": 0,
      "isDownloadProhibited": 0,
      "Integrity": {
        "md5": "",
        "sha1": "",
        "sha256": "",
        "sha384": ""
      },
      "FileData": {
        "Status": "none", "Timestamp": 0, "Message": "", "FileSize": 0, "FileModified": 0,
        "Duration": 0, "Bitrate": 0, "SampleRate": 0, "Channels": 0,
        "LUFS": null, "LRA": null, "TruePeak": null, "LeftRMS": null, "RightRMS": null,
        "ChannelDiff": null, "StereoSeparation": null, "DR": null,
        "WaveformURL": "", "ID3": {}, "CoverArt": 0, "Codec": ""
      },
      "ArtistTotalPlays": 6789,
      "NextUp": [
        {
          "Artist": "SoundLogic",
          "Title": "Monty on the run",
          "AddedDate": "2024-09-15 14:12:50",
          "CompositeRating": "5.0",
          "Voters": "1",
          "Image": "https://radio.ericade.net/images/demoscene.png",
          "StationID": "1",
          "TrackerType": "Modern remix (demo scene)",
          "TrackID": "14925"
        }
      ],
      "Playlog": [
        {
          "timestamphr": "2026-05-16 13:57:48"
        },
        {
          "timestamphr": "2026-05-07 22:38:33"
        },
        {
          "timestamphr": "2026-04-28 16:05:51"
        }
      ]
    }
  ]
}
```

Next Up

Returns the station's upcoming queue on its own, without the surrounding track record. Equivalent to the `NextUp` array that Now Playing embeds, but small enough to poll separately.

`POST` `https://api.ericade.net/radio/v2/getnextup`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	The station to read the queue for. Defaults to 1 when omitted.	1 = ericade.radio 2 = Best of ericade.radio

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "StationID": 1
}
```

Response

Returns a `{"NextUp": [ ... ]}` object with up to three entries. Each entry has the same fields as `NextUp` in the Now Playing response.

> When the station is running a manual override (a live show or a special message), the queue is replaced by a single synthetic entry: artist `ericade.radio`, title `Back to normal broadcast`, tracker type `Normal broadcast`, `AddedDate` and ratings zeroed.

Example response

```
{
  "NextUp": [
    {
      "Artist": "SoundLogic",
      "Title": "Monty on the run",
      "AddedDate": "2024-09-15 14:12:50",
      "CompositeRating": "5.0",
      "Voters": "1",
      "Image": "https://radio.ericade.net/images/demoscene.png",
      "StationID": "1",
      "TrackerType": "Modern remix (demo scene)",
      "TrackID": "14925"
    }
  ]
}
```

Real-time alternative: the `next-up` event on the Station Data WebSocket fires whenever the queue changes, avoiding the need to poll.

Track by ID

Returns full metadata for a specific track. The response schema is identical to the Now Playing endpoint.

`POST` `https://api.ericade.net/radio/v2/songs/<trackid>/`

Replace `<trackid>` in the URL with the numeric `TrackID`. The path segment wins — a `TrackID` in the body is ignored.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	`Password` is accepted as an alias.
`StationID`	Yes	The station the track belongs to.	1 = ericade.radio 2 = Best of ericade.radio
`LastPlaysToReturn`	No	Number of `Playlog` entries to return. Defaults to 3, hard-capped at 15.	3
`ReturnTranscripts`	No	[Podcast] 0 = return an empty `Transcript` (default), 1 = return the full transcript text.	
`ReturnProductionNotes`	No	[Podcast] 0 = off (default), 1 = include `ProductionNotes`.	
`EligibilityFilter`	No	0 = return the track regardless (default), 1 = omit it when it cannot be requested right now.	
`Sequence`	No	Same change-detection mechanism as Now Playing. Rarely useful here.	

Fixed server-side: `ReturnArtistDescription`, `ReturnArtistLongDescription` and `ReturnPlayList` are forced to 1, so the artist biography and podcast chapter list are always included. `NowPlaying`, `ReturnNextUp` and `DisableBufferCompensation` are not applicable and are not forwarded.

Sending `Limit` switches this endpoint into paginated search mode, which ignores the track id in the path and returns a page of the whole library instead. Do not send `Limit` when you want one track.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1Qi … ",
  "StationID": 1,
  "LastPlaysToReturn": 3,
  "ReturnTranscripts": 1
}
```

Search

Searches the station's track library by a free-text term and returns matching tracks. The response schema is identical to the Now Playing response.

`POST` `https://api.ericade.net/radio/v2/search`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	The station to search.	1 = ericade.radio 2 = Best of ericade.radio
`Term`	Yes	Search string, matched against both title and artist, e.g. "my wolf". Prefix with `Tags:` to search the tag list instead.	my wolf Tags: Revision 2026
`Limit`	No	Maximum number of results to return. Defaults to 10 and is capped at 30.	20
`Skip`	No	Number of results to skip from the start (for pagination).	0
`ReturnPlayList`	No	[Podcast] 0 = off (default), 1 = include `PlayList`.	
`ReturnTranscripts`	No	[Podcast] 0 = return an empty `Transcript` (default), 1 = return the full transcript text.	

Fixed server-side: `ReturnArtistDescription`, `ReturnArtistLongDescription`, `ReturnProductionNotes` and `ReturnShowNotes` are forced to 1, and eligibility filtering is forced off, so every match is returned with its full artist biography. Search runs across the whole library — `StationID` is used for the station name and offsets in the response, not as a filter on which tracks match.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1Qi … ",
  "StationID": 1,
  "Term": "my wolf",
  "Limit": 20,
  "Skip": 0
}
```

Response

Returns a `{"Tracks": [ … ]}` object. Each entry has the same fields as the Now Playing response (Artist, Title, StationID, StationName, TrackCanBeRequested, RequestVerdict, TrackID, TrackerType, CompositeRating, Image, Media, Playlog, etc.). `NextUp` is omitted — it is not meaningful for search results.

Track Listing

Paged listing of the station's track library with a total count, for browse and report views. This endpoint only ever queries the `titles` table (no joins), so every field the Now Playing response has that is sourced from that table is present here too, under the same name. What is missing is exactly what a join or a separate table would be needed for: `TrackArtists`, `NextUp`, per-artist rating/bio/external-link fields, `Playlog`, `TimeStamp`/`TimeStampHumanReadable`, `isListenerRequested`, `isOverride` and `ArtistTotalPlays`. Use Track by ID when you need those.

`POST` `https://api.ericade.net/radio/v2/songs`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	The station to list tracks from.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	Number of tracks to return. Defaults to 10; values outside 0–9000 are rejected.	25
`Skip`	No	Number of tracks to skip from the start (for pagination). Values outside 0–10000 are rejected.	0
`Search`	No	Free-text term matched against artist and title. Prefix with `Tag:` to search the tag list instead. Max 30 characters. When omitted, station-ID jingles (artist `ericade.radio`) are excluded.	virgill Tag:Revision

> Results are ordered by artist then title. On StationID 2 they are ordered by creation date descending instead.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "StationID": 1,
  "Skip": 0,
  "Limit": 25,
  "Search": "virgill"
}
```

Response

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Query ⇒ ResultCount`	Number of tracks in this response.	25
`Query ⇒ TotalCount`	Total number of tracks matching the query, ignoring `Skip`/`Limit`.	312

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Tracks ⇒ ID`, `TrackID`, `TitleID`	Unique track identifier. All three are the same value — `ID` is this endpoint's original key, `TrackID`/`TitleID` are added for parity with Now Playing.	10147
`Tracks ⇒ StationID`	The station this track belongs to.	1
`Tracks ⇒ StationName`	Human-readable station name.	24/7 tracked music
`Tracks ⇒ WebStreamingOffset`	Web streaming offset for the station in seconds.	12
`Tracks ⇒ StreamingOffset`	Streaming offset for the station in seconds.	10
`Tracks ⇒ Title`	Track title.	Bratgrumbeere
`Tracks ⇒ Fullartist`, `Artist`	Complete artist field including all collaborators. Both keys carry the same value; `Fullartist` is this endpoint's original key, `Artist` matches Now Playing.	Virgill
`Tracks ⇒ Slug`	URL-friendly slug for the track.	https://ericade.radio/song/10147/bratgrumbeere-virgill
`Tracks ⇒ TrackUpdatedAt`	When the track's metadata was last updated.	2026-06-12 09:41:02
`Tracks ⇒ Album`	Album tag from the audio file's metadata.	Original Amiga Works (.mod) TERN-Okt2020-S1
`Tracks ⇒ Image`	Image URL derived from the album/format.	https://radio.ericade.net/images/mod.png
`Tracks ⇒ Creationdatehr`, `CreationDateHR`, `AddedDate`	When the track was first added, human-readable. All three carry the same value; `Creationdatehr` is this endpoint's original key, the other two match Now Playing (`AddedDate` is deprecated there too).	2020-10-14 12:02:11
`Tracks ⇒ CreationDate`	When the track was first added, Unix epoch string.	1602676931
`Tracks ⇒ TrackCanBeStarred`	Whether this track can receive a star rating right now.	0 or 1
`Tracks ⇒ Length`, `PlayLength`	Length between Cue In and Cue Out, in seconds. Same value under both keys; `Length` is a string (this endpoint's original key), `PlayLength` is a number matching Now Playing.	292.757
`Tracks ⇒ Lengthhr`, `PlayLengthHR`	Length expressed as MM:SS.	04:52
`Tracks ⇒ Batch`	Batch number auto-parsed from the Album field.	TERN-Okt2020-S1
`Tracks ⇒ Tracktype`, `TrackerType`	Type/format of the track. Same value under both keys; `Tracktype` is this endpoint's original key, `TrackerType` matches Now Playing.	Amiga 4-channel module
`Tracks ⇒ Type`	Stated/inferred song type from PlayIT Live. Not actively used.	Song
`Tracks ⇒ isPodcast`	Whether this is a podcast episode rather than a music track.	0 or 1
`Tracks ⇒ Podcast`	Present in all responses. Empty string for non-podcast tracks.	""
`Tracks ⇒ PodcastURL`	[Podcast] URL to the podcast audio file.	
`Tracks ⇒ EpisodeNumber`	[Podcast] Episode number. Empty for non-podcast tracks.	
`Tracks ⇒ BroadcastDate`	[Podcast] Broadcast date of the podcast episode.	

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Tracks ⇒ About`	Short description from the song or podcast episode.	
`Tracks ⇒ ProductionNotes`	[Podcast] Production notes. Empty for non-podcast tracks.	
`Tracks ⇒ PlayList`	[Podcast] Chapter list for the podcast episode. Empty for non-podcast tracks.	
`Tracks ⇒ Transcript`	[Podcast] Full transcript of the episode. Empty for non-podcast tracks.	
`Tracks ⇒ Footer`	[Podcast] Footer text/show notes appended to the episode. Empty for non-podcast tracks.	
`Tracks ⇒ Equipment`	[Podcast] Equipment used to record the episode. Empty for non-podcast tracks.	
`Tracks ⇒ CompositeRating`	Track rating from 0.0–5.0. Higher is better.	4.3
`Tracks ⇒ Votes`	Number of votes the track has received.	2
`Tracks ⇒ TrackTotalPlays`	Number of times this track has been played on the station.	313
`Tracks ⇒ NowPlayingOnPlaylist`	[Podcast] Chapter that would be current right now, computed the same way as on Now Playing.	
`Tracks ⇒ TrackCanBeRequested`	Whether this track can be requested right now. Computed the same way as Now Playing, except it does not check the next-up queue (that needs the `nextup` table), so a track that is next in queue is not flagged ineligible for that reason here.	0 or 1
`Tracks ⇒ RequestVerdict`	Human-readable reason from the eligibility check above.	Song was played recently
`Tracks ⇒ isLive`	Whether the Album field marks this as a live recording.	0 or 1
`Tracks ⇒ isRemote`	Whether the Type field marks this as a remote broadcast.	0 or 1
`Tracks ⇒ isNew`	Whether the track was added within the station's new-track window.	0 or 1
`Tracks ⇒ NewDays`	Number of days a track is considered new on this station.	7
`Tracks ⇒ isBroadcastProhibited`	Whether broadcasting this track is prohibited.	0 or 1
`Tracks ⇒ isDownloadProhibited`	Whether downloading this track is prohibited. If 1, the music URLs and every `Media` entry are empty.	0 or 1
`Tracks ⇒ musicURL`	Direct URL to the track's audio file.	https://radio.ericade.net/mods/virgill-bratgrumbeere.mod
`Tracks ⇒ musicURLFlac`	Direct URL to the track's FLAC audio file, when available. Empty if no FLAC version exists.	
`Tracks ⇒ musicURLOriginal`	*(Not yet implemented) Always an empty string — see the note on this field in the Now Playing table.*	
`Tracks ⇒ Media`	The three URLs above expressed as a typed list, in the fixed order mp3, flac, original.	Array[]

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Tracks ⇒ Integrity`	Checksums of the track's audio file: `md5`, `sha1`, `sha256` and `sha384`. Same object as `Integrity` in the Now Playing response — see that table. Each value is an empty string for a track that has not been checksummed.	Object{}
`Tracks ⇒ FileData`	What was measured in the track's audio file, and the address of its waveform picture. Same object as `FileData` in the Now Playing response — see that table. `Status` is `none` for a track that has not been scanned, which at the moment is every track that is not a podcast episode.	Object{}
`Tracks ⇒ Path`	Internal playout path of the file.	
`Tracks ⇒ Guid`	Internal GUID from PlayIT Live. Not intended for public use.	aace857b-366a-46fe-9208-1d5b9fc0caae
`Tracks ⇒ Tags`	Comma-separated tag list.	Revision, Revision 2026
`Tracks ⇒ TagList`	List of tags associated with the track.	[]
`Tracks ⇒ MA_NextUpdate`	(Kept for backward compatibility) Same value as `ModArchiveEnrichment ⇒ MA_NextUpdate` below.	
`Tracks ⇒ ModArchiveTrueFileName`	The module's real filename as recorded on ModArchive.org. See the Now Playing table for the full description.	virgill-bratgrumbeere.mod
`Tracks ⇒ ModArchiveEnrichment`	Module metadata from ModArchive.org. Same shape as `ModArchiveEnrichment` in the Now Playing response — see that table for the full `MA_*` field list.	Array[]

Fields not listed here (`TrackArtists`, `NextUp`, per-artist rating/bio/external-link fields, `Playlog`, `TimeStamp`, `TimeStampHumanReadable`, `isListenerRequested`, `isOverride`, `ArtistTotalPlays`) are not present on this endpoint — they need a join or a separate table, and this endpoint only ever queries `titles`. Use Track by ID for those.

When nothing matches, the response is `{ "Software":["No track information was found."]}` with HTTP 200 — there is no `Query` or `Tracks` key at all. Handle that shape explicitly.

Example response


```
{
  "Query": [
    {
      "ResultCount": "1",
      "TotalCount": "312"
    }
  ],
  "Tracks": [
    {
      "ID": 10147,
      "TrackID": 10147,
      "TitleID": 10147,
      "StationID": 1,
      "StationName": "24/7 tracked music",
```

```
"WebStreamingOffset": 12,
"StreamingOffset": 10,
"Title": "Bratgrumbeere",
"Fullartist": "Virgill",
"Artist": "Virgill",
"Slug": "https://ericade.radio/song/10147/bratgrumbeere-virgill",
"TrackUpdatedAt": "2026-06-12 09:41:02",
"Album": "Original Amiga Works (.mod) TERN-Okt2020-S1",
"Image": "https://radio.ericade.net/images/mod.png",
"Creationdatehr": "2020-10-14 12:02:11",
"CreationDateHR": "2020-10-14 12:02:11",
"CreationDate": "1602676931",
"AddedDate": "2020-10-14 12:02:11",
"TrackCanBeStarred": 1,
"Length": "292.757",
"Lengthhr": "04:52",
"PlayLength": 292.757,
"PlayLengthHR": "04:52",
"Batch": "TERN-Okt2020-S1",
"Tracktype": "Amiga 4-channel module",
"TrackerType": "Amiga 4-channel module",
"Type": "Song",
"isPodcast": 0,
"Podcast": "",
"PodcastURL": "",
"EpisodeNumber": "",
"BroadcastDate": "",
"About": "",
"ProductionNotes": "",
"PlayList": "",
"Transcript": "",
"Footer": "",
"Equipment": "",
"CompositeRating": 0,
"Votes": 0,
"TrackTotalPlays": 313,
"NowPlayingOnPlaylist": "",
"TrackCanBeRequested": 1,
"RequestVerdict": "Song can be requested",
"isLive": 0,
"isRemote": 0,
"isNew": 0,
"NewDays": 7,
"isBroadcastProhibited": 0,
"isDownloadProhibited": 0,
"musicURL": "https://radio.ericade.net/mods/virgill-bratgrumbeere.mod",
"musicURLFlac": "",
"musicURLOriginal": "",
"Media": [
  { "type": "mp3", "url": "https://radio.ericade.net/mods/virgill-bratgrumbeere.mod" },
  { "type": "flac", "url": "" },
  { "type": "original", "url": "" }
],
"Integrity": {
  "md5": "",
  "sha1": "",
  "sha256": "",
  "sha384": ""
```

```
      },
      "FileData": {
        "Status": "none", "Timestamp": 0, "Message": "", "FileSize": 0, "FileModified": 0,
        "Duration": 0, "Bitrate": 0, "SampleRate": 0, "Channels": 0,
        "LUFS": null, "LRA": null, "TruePeak": null, "LeftRMS": null, "RightRMS": null,
        "ChannelDiff": null, "StereoSeparation": null, "DR": null,
        "WaveformURL": "", "ID3": {}, "CoverArt": 0, "Codec": ""
      },
      "Path": "\\mods\\virgill-bratgrumbeere.mod",
      "Guid": "aace857b-366a-46fe-9208-1d5b9fc0caae",
      "Tags": "",
      "TagList": [],
      "MA_NextUpdate": "",
      "ModArchiveTrueFileName": "virgill-bratgrumbeere.mod",
      "ModArchiveEnrichment": [
        {
          "MA_Format": "",
          "MA_URL": "",
          "MA_Date": "",
          "MA_Hash": "",
          "MA_Size": "",
          "MA_SongTitle": "",
          "MA_Instruments": "",
          "MA_GenreID": 0,
          "MA_Rating": 0,
          "MA_Comments": "",
          "MA_ReviewRating": 0,
          "MA_ReviewTotal": 0,
          "MA_ImageUrl": "",
          "MA_Artists": "",
          "MA_GuessedArtists": "",
          "MA_LastUpdated": 0,
          "MA_HasFile": 0,
          "MA_Bytes": 0,
          "MA_TimeStamp": 0,
          "MA_Found": 0,
          "MA_LastSync": "",
          "MA_NextUpdate": ""
        }
      ]
    }
  ]
}
```

Artists

Paged listing of artists, each with their biography, external links, rating and the complete list of tracks credited to them.

`POST` `https://api.ericade.net/radio/v2/artists`

Response size. Every artist carries a nested `Tracks` array of *full* track objects, so a page of 30 prolific artists can run to several megabytes. `Limit` is capped at 30 for that reason. Use Sitemap Artists when you only need the flat artist records.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	Validated against the station configuration, but does not filter the result set — artist records are not station-scoped.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	Number of artists to return. Defaults to 10; more than 30 is rejected.	10
`Skip`	No	Number of artists to skip from the start (for pagination). Values outside 0–10000 are rejected.	0
`Search`	No	Free-text term matched against the artist name. Max 30 characters. When omitted, the `ericade.radio` station identity is excluded.	huelsbeck

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 1,
  "Skip": 0,
  "Limit": 10,
  "Search": "huelsbeck"
}
```

Response

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Query ⇒ ResultCount`	Number of artists in this response.	1
`Query ⇒ TotalCount`	Total number of artists matching the query.	1
`Artists ⇒ ArtistID`	Unique artist identifier. Matches `TrackArtists ⇒ ArtistID` on the track endpoints.	4821
`Artists ⇒ Artist`	Artist name.	Chris Huelsbeck

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Artists ⇒ ShortDescription`	Short artist bio.	(text)
`Artists ⇒ LongDescription`	Full artist bio.	(text)
`Artists ⇒ Image`	The artist's picture: a complete `https://` address, or an empty string when the artist has none. Always present. For a picture uploaded through artistdata/image/add the same name exists as `.webp` and, scaled, as `-150x150`, `-300x300`, `-768x768` and `-1024x1024` — see Pictures of songs and artists. Unlike the texts around it, it is handed out exactly as stored.	https://radio.ericade.net/images/artists/4821.png
`Artists ⇒ CompositeRating`	Artist rating from 0.0–5.0.	4.6
`Artists ⇒ Votes`	Number of votes the artist has received.	12
`Artists ⇒ TotalPlays`	Number of times this artist has been played.	6789
`Artists ⇒ Album`	*(Not populated) Always an empty string.*	
`Artists ⇒ Demozoo`, `Wikipedia`, `CSDB`, `OtherUrl`, `ModArchive`, `Bandcamp`, `SoundCloud`, `YouTube`, `Pouet`	External links for the artist.	https://demozoo.org/sceners/7637/
`Artists ⇒ isBroadcastProhibited`	Whether broadcasting this artist's tracks is prohibited.	0 or 1
`Artists ⇒ isDownloadProhibited`	Whether downloading this artist's tracks is prohibited.	0 or 1
`Artists ⇒ Tracks`	Every track credited to this artist, each a full track object with the same fields as the Now Playing response.	Array[]

When nothing matches, the response is `{ "Tracks":["No track information was found."]}` — note both the unexpected key (`Tracks`, not `Artists`) and the array-of-strings shape. Handle it explicitly.

Example response


```
{
  "Query": [
    {
      "ResultCount": "1",
      "TotalCount": "1"
    }
  ],
  "Artists": [
    {
      "ArtistID": 4821,
      "Artist": "Chris Huelsbeck",
      "ShortDescription": "A true legend on the Amiga and other systems.",
      "LongDescription": "A true legend on the Amiga and other systems. He's most famous for the music
from Turrican I, II and III.",
      "Image": "https://radio.ericade.net/images/artists/4821.png",
      "CompositeRating": 4.6,
      "Votes": 12,
```

```
      "TotalPlays": 6789,
      "Album": "",
      "Demozoo": "https://demozoo.org/sceners/7637/",
      "Wikipedia": "",
      "CSDB": "",
      "OtherUrl": "",
      "ModArchive": "",
      "Bandcamp": "https://chrishuelsbeck.bandcamp.com/",
      "SoundCloud": "",
      "YouTube": "",
      "Pouet": "",
      "isBroadcastProhibited": 0,
      "isDownloadProhibited": 0,
      "Tracks": [
        {
          "Artist": "Chris Huelsbeck",
          "Title": "Dressed to chill",
          "TrackID": 14051,
          "TrackerType": "Modern remix",
          "CompositeRating": 0,
          "Votes": 0,
          "Image": "https://radio.ericade.net/images/rmx.png"
        }
      ]
    }
  ]
}
```


The `Tracks` entries are abbreviated above. In a real response each is a full track record with the same field set as the Now Playing response — see that table for every field. Nested track objects always carry `NextUp` as an empty array; the upcoming queue is only meaningful on the now-playing endpoints.

Artist by ID

Returns one artist. The response schema is identical to the Artists endpoint.

`POST` `https://api.ericade.net/radio/v2/artists/<artistid>`

Replace `<artistid>` with the numeric `ArtistID`, which you can obtain from the `TrackArtists` array on any track record.

Request fields are the same as Artists (`Token`, `StationID`, optional `Limit` and `Skip`).

> Sending a `Search` value overrides the path segment: the search takes precedence and the artist id is ignored. Send one or the other, not both.

Sitemap Artists

Exports artists as flat records without the nested track list, plus the last-modified timestamp a sitemap needs. Because nothing is nested, `Limit` goes up to 10000 here rather than the 30 of the Artists endpoint.

`POST` `https://api.ericade.net/radio/v2/sitemapartists`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	Validated against the station configuration, but does not filter the result set.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	Number of artists to return. Defaults to 10; more than 10000 is rejected.	5000
`Skip`	No	Number of artists to skip from the start (for pagination). Values outside 0–10000 are rejected.	0
`Search`	No	Free-text term matched against the artist name. Max 30 characters.	

> Despite the path shape, this endpoint ignores any trailing id segment and always returns a page of artists.

Response

Returns the same `Query` counters as the other list endpoints, followed by an `Artists` array.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`ArtistID`	Unique artist identifier.	4821
`Artist`	Artist name.	Chris Huelsbeck
`ShortDescription` / `LongDescription`	Artist bio.	(text)
`TotalPlays`	Number of times this artist has been played.	6789
`LastPlayed`	Unix epoch of the artist's most recent play.	1780262202
`CompositeRating` / `Votes`	Artist rating and vote count.	4.6 / 12
`Demozoo`, `Wikipedia`, `CSDB`, `OtherUrl`, `ModArchive`, `Bandcamp`, `SoundCloud`, `YouTube`, `Pouet`	External links for the artist.	
`Guid`	Internal identifier. Not intended for public use.	
`EligibilityTime`	Request-eligibility timer for the artist, Unix epoch.	1780262202
`ArtistUpdatedAt`	When the artist record was last updated. This is the value to use as `<lastmod>` in a sitemap.	2026-06-12 09:41:02

Example response


```
{
  "Query": [
    {
      "ResultCount": "1",
      "TotalCount": "5312"
    }
  ],
  "Artists": [
    {
      "ArtistID": 4821,
      "Artist": "Chris Huelsbeck",
      "ShortDescription": "A true legend on the Amiga and other systems.",
      "LongDescription": "A true legend on the Amiga and other systems.",
      "TotalPlays": 6789,
      "LastPlayed": "1780262202",
      "CompositeRating": "4.6",
      "Votes": "12",
      "Demozoo": "https://demozoo.org/sceners/7637/",
      "Wikipedia": "",
      "CSDB": "",
      "OtherUrl": "",
      "ModArchive": "",
      "Bandcamp": "https://chrishuelsbeck.bandcamp.com/",
      "SoundCloud": "",
      "YouTube": "",
      "Pouet": "",
      "Guid": "",
      "EligibilityTime": "1780262202",
      "ArtistUpdatedAt": "2026-06-12 09:41:02"
    }
  ]
}
```

Song History

Returns the recently played tracks on a station.

`POST` `https://api.ericade.net/radio/v2/songhistory`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint. `Password` is accepted as an alias.
`StationID`	Yes	The station to get history for.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	Number of history entries to return. Defaults to 10; more than 9000 is rejected.	8
`LastAddedMode`	No	0 = recently played, newest first (default). 1 = recently added to the library, newest first.	

> The default mode skips the single most recent play, so the first entry is the track before the one currently on air. That keeps the history list from duplicating what Now Playing already shows.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "StationID": 1,
  "Limit": 8
}
```

Response

Returns a `{"Tracks": [ ... ]}` array ordered most-recent first.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`TrackID`	Unique, stable track identifier.	10147
`TimeStampHR`	When this track started playing, as a human-readable date string.	2026-03-30 20:44:32
`TimeStamp`	When this track started playing, as a Unix epoch.	1774903472
`Artist`	Complete artist field.	Virgill
`Title`	Track title.	Bratgrumbeere
`TrackArtists`	Array of all artists linked to this track. Same object structure as `TrackArtists` in the Now Playing response.	Array[]

ATTRIBUTE	DESCRIPTION	EXAMPLE
`CreationDate`	When the track was first added, Unix epoch. Populated in last-added mode; empty in the default mode.	
`CreationDateHR`	(Not implemented) Always an empty string — the value is fetched but never written to the response.	
`Album`	Album tag from audio file metadata.	Original Amiga Works (.mod) TERN-Okt2020-S1
`Duration`	Duration of the track in seconds.	292.75718820861700000000
`TrackerType`	Type of tracker used for the track.	Amiga 4-channel module
`musicURL`	Direct URL to the track's audio file. Empty when downloading the track is prohibited. Song history returns only this one audio URL — there is no `Media` array or FLAC/original variant here.	https://radio.ericade.net/mods/dr_awesome-gone_for_good.mod
`Image`	URL to the track's image.	https://radio.ericade.net/images/mod.png
`IsListenerRequested`	Indicates if the track was requested by a listener (0 = no, 1 = yes).	0
`StationID`	The station this track belongs to.	1
`Guid`	(Will be removed) Internal GUID from PlayIT Live. Not intended for public use.	6894ac27-abc9-4c4a-aad2-15879dda4dff

Example response


```
{
    "Tracks": [
        {
            "TrackID": "10147",
            "TimeStampHR": "2026-05-31 21:16:42",
            "TimeStamp": "1780262202",
            "Artist": "Dr. Awesome",
            "Title": "Gone For Good",
            "TrackArtists": [
                {
                    "ArtistID": 5133,
                    "Artist": "Dr. Awesome",
                    "ShortDescription": "",
                    "LongDescription": "",
                    "CompositeRating": 4.2,
                    "Votes": 8,
                    "TotalPlays": 412,
                    "Demozoo": "",
                    "Wikipedia": "",
                    "CSDB": "",
                    "OtherUrl": "",
                    "ModArchive": "",
                    "Bandcamp": "",
                    "SoundCloud": "",
                    "YouTube": "",
                    "Pouet": "",
                    "isBroadcastProhibited": 0,
                    "isDownloadProhibited": 0
```

```
                }
            ],
            "CreationDate": "",
            "CreationDateHR": "",
            "Album": "Original Amiga Works (.mod) TERN-Okt2020-S1",
            "Duration": "292.75718820861700000000",
            "TrackerType": "Amiga 4-channel module",
            "musicURL": "https://radio.ericade.net/mods/dr_awesome-gone_for_good.mod",
            "Image": "https://radio.ericade.net/images/mod.png",
            "IsListenerRequested": "0",
            "StationID": "1",
            "Guid": "aace857b-366a-46fe-9208-1d5b9fc0caae"
        },
        {
            "TrackID": "12115",
            "TimeStampHR": "2026-05-31 21:16:34",
            "TimeStamp": "1780262194",
            "Artist": "ericade.radio",
            "Title": "You want to hear your favorite song on this station. Go to ericade.radio-song",
            "TrackArtists": [],
            "CreationDate": "",
            "CreationDateHR": "",
            "Album": "Format:.stn TERN.",
            "Duration": "10.44897959183670000000",
            "TrackerType": "StationID",
            "musicURL": "https://radio.ericade.net/mods/ericade_stationid_12115.stn",
            "Image": "https://radio.ericade.net/images/stn.png",
            "IsListenerRequested": "0",
            "StationID": "1",
            "Guid": "3933ca49-d5f3-426e-bbfd-494695977262"
        }
    ]
}
```

Playout

Returns either the recently played tracks or the recently added ones, selected by `LastAddedMode`. This is the older, differently shaped sibling of Song History — it carries rating and request-eligibility fields that song history does not.

`POST` `https://api.ericade.net/radio/v2/playout`

Use `LastAddedMode: 1`. In the default mode the underlying query does not select the creation-date columns, and the endpoint emits `"CreationDate":` with no value — strict JSON parsers reject the body. Until that is fixed, use last-added mode here, or use Song History for play history.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	The station to read from.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	Number of entries to return. Defaults to 10; more than 100 is rejected.	20
`LastAddedMode`	No	0 = recently played, newest first (default — see the warning above). 1 = recently added to the library, newest first, with the rating and eligibility fields included.	1

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "StationID": 1,
  "Limit": 20,
  "LastAddedMode": 1
}
```

Response

Returns a `{"Tracks": [ ... ]}` object. Fields marked [LastAddedMode=1] are present only in last-added mode.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`TimeStampHR`	When the track played, or was added in last-added mode, human-readable.	2026-03-30 20:44:32
`TimeStamp`	The same moment as a Unix epoch.	1774903472
`Artist`	Complete artist field.	Dr. Awesome
`Title`	Track title.	Gone For Good

ATTRIBUTE	DESCRIPTION	EXAMPLE
`TrackArtists`	Array of all artists linked to this track. Same object structure as `TrackArtists` in the Now Playing response.	Array[]
`PlayLengthHR`	[LastAddedMode=1] Play length as MM:SS.	04:52
`PlayLength`	[LastAddedMode=1] Duration in seconds.	292.757
`TrackerType`	[LastAddedMode=1] Type/format of the track.	Amiga 4-channel module
`CompositeRating`	[LastAddedMode=1] Track rating from 0.0–5.0.	4.2
`Votes`	[LastAddedMode=1] Number of votes the track has received.	8
`TrackTotalPlays`	[LastAddedMode=1] Number of times this track has been played.	313
`TrackCanBeRequested`	[LastAddedMode=1] Whether the track can be requested right now.	0 or 1
`RequestVerdict`	[LastAddedMode=1] Human-readable reason why it cannot be requested. Echo this string to the user.	Song was played recently
`TrackID`	Unique, stable track identifier.	10147
`CreationDate`	[LastAddedMode=1] When the track was first added, Unix epoch.	1667315948
`CreationDateHR`	[LastAddedMode=1] The same moment, human-readable.	2022-11-01 16:19:08
`Album`	[LastAddedMode=1] Album tag from the audio file's metadata.	Original Amiga Works (.mod) TERN-Okt2020-S1
`Image`	URL to the track's image.	https://radio.ericade.net/images/mod.png
`musicURL`	Direct URL to the track's audio file.	https://radio.ericade.net/mods/dr_awesome-gone_for_good.mod
`StationID`	The station this track belongs to.	1
`Guid`	Internal GUID from PlayIT Live. Not intended for public use.	aace857b-366a-46fe-9208-1d5b9fc0caae

Example response


```
{
  "Tracks": [
    {
      "TimeStampHR": "2026-05-31 21:16:42",
      "TimeStamp": 1780262202,
      "Artist": "Dr. Awesome",
      "Title": "Gone For Good",
      "TrackArtists": [
        {
          "ArtistID": 5133,
          "Artist": "Dr. Awesome",
          "ShortDescription": "",
          "LongDescription": "",
          "CompositeRating": 4.2,
```

```
          "Votes": 8,
          "TotalPlays": 412,
          "Demozoo": "",
          "Wikipedia": "",
          "CSDB": "",
          "OtherUrl": "",
          "ModArchive": "",
          "Bandcamp": "",
          "SoundCloud": "",
          "YouTube": "",
          "Pouet": "",
          "isBroadcastProhibited": 0,
          "isDownloadProhibited": 0
        }
      ],
      "PlayLengthHR": "04:52",
      "PlayLength": 292.757,
      "TrackerType": "Amiga 4-channel module",
      "CompositeRating": 4.2,
      "Votes": 8,
      "TrackTotalPlays": 313,
      "TrackCanBeRequested": 1,
      "RequestVerdict": "Track can be requested",
      "TrackID": 10147,
      "CreationDate": 1667315948,
      "CreationDateHR": "2022-11-01 16:19:08",
      "Album": "Original Amiga Works (.mod) TERN-Okt2020-S1",
      "Image": "https://radio.ericade.net/images/mod.png",
      "musicURL": "https://radio.ericade.net/mods/dr_awesome-gone_for_good.mod",
      "StationID": 1,
      "Guid": "aace857b-366a-46fe-9208-1d5b9fc0caae"
    }
  ]
}
```

New Songs

Returns the tracks most recently added to the station's library, newest first, together with their ratings and current request eligibility. Use this to build a "new on the station" panel.

`POST` `https://api.ericade.net/radio/v2/new-songs`

This is the last-added view of Playout with the mode fixed server-side, so every entry always carries the rating and eligibility fields and the malformed-JSON caveat on that endpoint does not apply here.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	The station to read from.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	Number of tracks to return. Defaults to 10; more than 100 is rejected.	20

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 1,
  "Limit": 20
}
```

Response

Identical to the Playout response in last-added mode.

> To tell whether a track is new enough to badge in a player, prefer the `isNew` and `NewDays` fields on the Now Playing record — they reflect the station's own definition of "new".

News & Podcasts

This is the *reading* side, and it is public. News items are written through News Administration, episodes through Podcast; people comment on both through Comment. The `News` field holds HTML that has passed the allow-list when it was saved through the API.

Lists news posts and podcast episodes, with pagination, free-text and tag search, and optional transcripts. This is the endpoint behind the podcast display pages and the RSS feed builder.

`POST` `https://api.ericade.net/radio/v2/news`

This endpoint reads the JWT from `Password`, not `Token`. The by-id variant reads `Token`. Sending the wrong field leaves the call unauthenticated.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Password`	Yes	JWT token that authenticates the call.	Note the field name — not `Token`.
`StationID`	Yes	The station to read from.	1 = ericade.radio 2 = Best of ericade.radio
`Podcasts`	No	0 = do not filter on podcasts (default), 1 = only podcast entries, 2 = only podcast entries including the `Length` and `LengthHR` columns.	2
`News`	No	0 = off (default), 1 = include news entries. Combine with `Podcasts` to get both kinds in one response.	
`ID`	No	Return only the entry linked to this `TrackID`. Also switches on the length columns.	13770
`Limit`	No	Number of entries to return. Defaults to 200; values outside 0–200 are rejected.	20
`Skip`	No	Number of entries to skip from the start (for pagination). Values outside 0–300 are rejected.	0
`Search`	No	Free-text term matched against artist and title. Prefix with `Tag:` to search the tag list instead. Max 30 characters.	Tag:Revision 2026
`IncludeTags`	No	0 = off (default), 1 = join in the tag column so `Tags` and `TagList` are populated.	
`ReturnTranscripts`	No	0 = return an empty `Transcript` (default), 1 = return the full transcript text. Transcripts are large — leave this off unless you display them.	

Results are ordered by broadcast date descending, then episode number descending, then timestamp descending, then title.

Example request

```
{
  "Password": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 2,
  "Podcasts": 2,
  "Limit": 10,
  "Skip": 0,
  "IncludeTags": 1,
  "ReturnTranscripts": 0
}
```

Response

Returns the `Query` paging counters followed by a `Tracks` array. Fields marked [Podcast] are empty for news posts.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Query ⇒ ResultCount`	Number of entries in this response.	10
`Query ⇒ TotalCount`	Total number of entries matching the query.	97
`TrackID`	Identifier of the track the post is linked to.	13770
`Title`	Episode or post title.	Flashback episode 97
`Artist`	Show or author name.	ericade.radio
`TrackUpdatedAt`	When the entry was last updated.	2026-06-12 09:41:02
`About`	Short description of the episode or post.	
`ModArchiveTrueFileName`	The module's real filename as recorded on ModArchive.org — see the Now Playing table. Always empty for news posts, which are stored in a table that has no such column, and empty for podcast episodes, which are not tracker modules.	
`Slug`	URL-friendly slug for the entry.	https://radio.ericade.net/#/song/13770/flashback-97
`Episode`	[Podcast] Episode number as a string.	97
`Explicit`	Explicit-content flag for the podcast feed.	0 or 1
`LastBuild`	Timestamp of the last RSS feed build. Only present when the stats row exists.	2026-08-01 04:00:02
`Length`	Track length in seconds. Only present when `Podcasts=2` or `ID` is set.	3612.4
`LengthHR`	Track length as MM:SS (HH:MM:SS when over an hour). Same condition as `Length`.	01:00:12
`BroadcastDate`	[Podcast] Broadcast date of the episode.	2022-07-16
`Image`	Image URL for the entry.	https://radio.ericade.net/wp-content/uploads/2020/12/amiga_flashback.jpg
`PodcastImage`	[Podcast] Dedicated podcast artwork URL, when set.	
`CompositeRating`	Rating from 0.0–5.0.	4.3

ATTRIBUTE	DESCRIPTION	EXAMPLE
`TrackVotes`	Number of votes the entry has received. Note the name — it is `Votes` on the track endpoints.	12
`Tags`	Comma-separated tag list. Empty unless `IncludeTags=1` or a tag search was performed.	Amiga, Flashback
`TagList`	The same tags as an array.	[]
`Equipment`	[Podcast] Equipment used to record the episode.	
`PlayList`	[Podcast] Chapter list for the episode.	00:00 Introduction…
`ProductionNotes`	[Podcast] Production notes.	
`hasTranscript`	1 when a transcript exists. Use it to decide whether a second call with `ReturnTranscripts=1` is worth making.	0 or 1
`Transcript`	Full transcript text. Empty unless `ReturnTranscripts=1`.	
`ShowNotes`	[Podcast] Show notes for the episode.	
`Footer`	[Podcast] Footer text appended to the episode.	
`isNews`	1 when the entry is a news post.	0 or 1
`isPodcast`	1 when the entry is a podcast episode.	0 or 1
`isRSS`	1 when the entry is published in the RSS feed.	0 or 1
`News`	Body text of the news post.	
`PodcastFileLength`	[Podcast] Size of the audio file in bytes, for the RSS enclosure.	57829120
`PodcastURL`	[Podcast] URL to the podcast audio file.	https://radio.ericade.net/Flashback/97.mp3
`Integrity`	[Podcast] Checksums of the MP3 at `PodcastURL`: `md5`, `sha1`, `sha256` and `sha384`. Same object as `Integrity` in the Now Playing response. Each value is an empty string for news posts, and for an episode that has not been checksummed yet.	Object{}
`FileData`	[Podcast] What Scan podcast measured in the MP3 — length, loudness, channel balance, ID3 tags — and `WaveformURL`, the address of its waveform chart. Same object as `FileData` in the Now Playing response. `Status` is `none`, and everything else empty, for news posts and for an episode that has not been scanned.	Object{}
`EpisodeNumber`	[Podcast] Episode number as a number. 0 for news posts.	97

The `md5`, `sha256` and `sha384` checksums are also published as files next to each episode's MP3. For `https://radio.ericade.net/Flashback/97.mp3` they are `97.md5`, `97.sha256` and `97.sha384` in the same directory. Each file holds one line in the format `md5sum -c`, `sha256sum -c` and `sha384sum -c` read, so a downloaded episode can be checked by running, for example, `sha256sum -c 97.sha256` in the directory it was saved to.

When nothing matches, the response is `{ "Tracks":["No news information was found."]}` — an array of strings rather than of objects. Handle that shape explicitly.

Example response


```
{
  "Query": [
    {
      "ResultCount": "1",
      "TotalCount": "97"
    }
  ],
  "Tracks": [
    {
      "TrackID": "13770",
      "Title": "Flashback episode 97",
      "Artist": "ericade.radio",
      "TrackUpdatedAt": "2026-06-12 09:41:02",
      "About": "A trip back to the Amiga demo scene of 1992.",
      "ModArchiveTrueFileName": "",
      "Slug": "https://radio.ericade.net/#/song/13770/flashback-episode-97-ericade-radio",
      "Episode": "97",
      "Explicit": "0",
      "LastBuild": "2026-08-01 04:00:02",
      "Length": "3612.4",
      "LengthHR": "01:00:12",
      "BroadcastDate": "2022-07-16",
      "Image": "https://radio.ericade.net/wp-content/uploads/2020/12/amiga_flashback.jpg",
      "PodcastImage": "",
      "CompositeRating": "4.3",
      "TrackVotes": "12",
      "Tags": "Amiga, Flashback",
      "TagList": ["Amiga", "Flashback"],
      "Equipment": "",
      "PlayList": "00:00 Introduction … ",
      "ProductionNotes": "",
      "hasTranscript": "1",
      "Transcript": "",
      "ShowNotes": "",
      "Footer": "",
      "isNews": "0",
      "isPodcast": "1",
      "isRSS": "1",
      "News": "",
      "PodcastFileLength": "57829120",
      "PodcastURL": "https://radio.ericade.net/Flashback/97.mp3",
      "Integrity": {
        "md5": "ab8effe6ef0c5eb07803dd92ac14ac09",
        "sha1": "803d89f5701d0882a97e43ebfb231928e7ca4522",
        "sha256": "b886baff39c8ed3489d9275aa1cbd24af9b1a1885b02a0789d69242af7ff4176",
        "sha384":
"f565bafcadb72fd55ef9948f1d63f2bf1ad060d02b40a63ca03209d125ec1abd730bedd0ee7c8947709953d029832160"
      },
      "FileData": {
        "Status": "ok", "Timestamp": 1789745361, "Message": "", "FileSize": 57829120, "FileModified":
1789700112,
        "Duration": 3614.302, "Bitrate": 128000, "SampleRate": 44100, "Channels": 2,
        "LUFS": -16.4, "LRA": 5.2, "TruePeak": -1.3, "LeftRMS": -19.1, "RightRMS": -19.7,
        "ChannelDiff": 0.6, "StereoSeparation": -14.8, "DR": 11.6,
        "WaveformURL": "https://radio.ericade.net/images/waveforms/97.webp?v=1789745361",
        "ID3": { "title": "97. Amiga &amp;amp; friends", "artist": "DJ Daemon", "genre": "Podcast" },
```

```
        "CoverArt": 1, "Codec": "mp3"
      },
      "EpisodeNumber": 97
    }
  ]
}
```

News by ID

Returns one news post or podcast episode. The response schema is identical to the News & Podcasts endpoint.

`POST` `https://api.ericade.net/radio/v2/news/<id>`

Replace `<id>` with the `TrackID` the entry is linked to. Addressing a single entry also switches on the `Length` and `LengthHR` columns.

Request fields are the same as News & Podcasts (`StationID`, optional `Podcasts`, `News`, `Limit`, `Skip`, `Search`, `ReturnTranscripts`), with one difference: **this route reads the JWT from `Token`**, not `Password`.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 2,
  "ReturnTranscripts": 1
}
```

Statistics

Returns various statistics for a station.

`POST` `https://api.ericade.net/radio/v2/stats`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	(Not implemented) Accepted, but the response always covers every station.	1 = ericade.radio 2 = Best of ericade.radio

`Limit` and `Days` are **not** forwarded on this path — the router only reads them on Statistics by FilterType. On this endpoint every top/bottom list is fixed at 10 rows and the rating and request windows are fixed at 30 days. Earlier revisions of this document listed both fields here as working; that was incorrect.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "StationID": 1
}
```

Response

Returns a `{"Stations": [ ... ]}` array. Each element in `Stations` represents one station and contains the fields below.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`StationID`	The station identifier.	1
`StationName`	Human-readable station name.	24/7 tracked music
`TotalLength`	Total combined length of all tracks on the station, in hours.	180.4
`TracksOnStation`	Total number of tracks on the station.	3125
`Trackerstats`	Array of tracker-type breakdowns. Each object contains `TrackerType` (name of the tracker format), `Tracks` (number of tracks of that type), `Percent` (percentage of total), and `StationID`.	Array[]
`Requests`	Array of the most-requested tracks within the last `Days` days. Each object is a full Track record (see Now Playing response) plus a `Count` field (number of times requested in the period).	Array[]

ATTRIBUTE	DESCRIPTION	EXAMPLE
`TopTracks`	Array of the highest-rated tracks within the last `Days` days. Each object is a full Track record (see Now Playing response).	Array[]
`BottomTracks`	Array of the lowest-rated tracks within the last `Days` days. Same structure as `TopTracks`.	Array[]
`TopArtists`	Array of the highest-rated artists. Each object contains `Artist`, `CompositeRating`, and `Voters`.	Array[]
`BottomArtists`	Array of the lowest-rated artists. Same fields as `TopArtists`.	Array[]
`StreamingCountries`	Array of listener countries ordered by session count. Each object contains `Country` (country name) and `Count` (number of streaming sessions).	Array[]
`StreamingAgents`	Array of user-agent strings seen in streaming sessions, ordered by count. Each object contains `Agent` and `Count`.	Array[]
`LogoffPlay`	Array of tracks most commonly playing when listeners disconnect. Each object contains `Track` (formatted "Artist-Title." string), `NumberOfPlays`, and `StationID`.	Array[]
`Duplicates`	Array of tracks that appear more than once in the library (same artist and title, different IDs). Each object contains `FullArtist`, `Title`, and `TrackID`.	Array[]

The `Requests`, `TopTracks` and `BottomTracks` entries in the example below are abbreviated for readability. In a real response each object is a full track record with the same field set as the Now Playing response (see that table for every field), plus a `Count` field on `Requests` entries.

Example response


```
{
    "Stations": [
        {
            "StationID": "1",
            "StationName": "24/7 tracked music",
            "TotalLength": "180.4",
            "TracksOnStation": "3125",
            "Trackerstats": [
                {
                    "TrackerType": "Amiga 4-channel
module",
                    "Tracks": "1180",
                    "Percent": "37.8",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Fasttracker",
                    "Tracks": "873",
                    "Percent": "27.9",
                    "StationID": "1"
                },
                {
                    "TrackerType":
"Impulsetracker",
                    "Tracks": "539",
                    "Percent": "17.2",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Modern remix",
                    "Tracks": "169",
                    "Percent": "5.4",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Screamtracker",
                    "Tracks": "138",
                    "Percent": "4.4",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Modern remix
```

```
(demo scene)",
                    "Tracks": "131",
                    "Percent": "4.2",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Multitracker",
                    "Tracks": "49",
                    "Percent": "1.6",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Pretracker",
                    "Tracks": "24",
                    "Percent": "0.8",
                    "StationID": "1"
                },
                {
                    "TrackerType": "OctaMed/MED",
                    "Tracks": "6",
                    "Percent": "0.2",
                    "StationID": "1"
                },
                {
                    "TrackerType": "AHX Synth-
tracker",
                    "Tracks": "4",
                    "Percent": "0.1",
                    "StationID": "1"
                },
                {
                    "TrackerType": "MO3",
                    "Tracks": "3",
                    "Percent": "0.1",
                    "StationID": "1"
                },
                {
                    "TrackerType": "C64 Sidtune",
                    "Tracks": "3",
                    "Percent": "0.1",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Digi Booster
Pro",
                    "Tracks": "2",
                    "Percent": "0.1",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Unknown",
                    "Tracks": "1",
                    "Percent": "0.0",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Digi Booster",
                    "Tracks": "1",
                    "Percent": "0.0",
                    "StationID": "1"
                },
                {
                    "TrackerType": "Scream tracker
2",
                    "Tracks": "1",
                    "Percent": "0.0",
                    "StationID": "1"
                },
                {
                    "TrackerType": "SoundFX /
MultiMedia Sound",
                    "Tracks": "1",
                    "Percent": "0.0",
                    "StationID": "1"
                }
            ],
            "Requests": [
                {
                    "Artist": "Ekorren",
                    "Title": "Little Danyjel",
                    "TrackID": "14050",
                    "Count": "20"
                },
                {
                    "Artist": "Synthesized World
2",
                    "Title": "Radio Smile",
                    "TrackID": "",
                    "Count": "20"
                },
                {
                    "Artist": "Drakonpod",
                    "Title": "Goa-ing",
                    "TrackID": "",
                    "Count": "16"
                },
                {
                    "Artist": "Maf",
                    "Title": "Mafland remix",
                    "TrackID": "14907",
                    "Count": "16"
                },
                {
                    "Artist": "Arpegiator of
Chryseis",
                    "Title": "My wolf 2",
                    "TrackID": "10663",
                    "Count": "13"
                },
                {
                    "Artist": "Yannis Brown",
                    "Title": "Tres Dilettantes",
                    "TrackID": "",
                    "Count": "12"
                },
                {
```

```
                    "Artist": "Xyce",
                    "Title": "Mirage",
                    "TrackID": "",
                    "Count": "11"
                },
                {
                    "Artist": "CyberZip",
                    "Title": "Ahh My Goddess!",
                    "TrackID": "13528",
                    "Count": "10"
                },
                {
                    "Artist": "Dr Vector",
                    "Title": "Auto rock",
                    "TrackID": "15441",
                    "Count": "9"
                },
                {
                    "Artist": "Arcane toaster",
                    "Title": "Fly Little Dino",
                    "TrackID": "",
                    "Count": "8"
                }
            ],
            "TopTracks": [
                {
                    "Artist": "Xyce",
                    "Title": "Rymdkraft - Dolph's
aerobics (remix)",
                    "CompositeRating": "5.0",
                    "Voters": "5",
                    "TrackID": "12003"
                },
                {
                    "Artist": "Nusrat",
                    "Title": "Summertime",
                    "CompositeRating": "5.0",
                    "Voters": "4",
                    "TrackID": "13229"
                },
                {
                    "Artist": "Trackerartist",
                    "Title": "Dragons lair",
                    "CompositeRating": "5.0",
                    "Voters": "4",
                    "TrackID": "10186"
                },
                {
                    "Artist": "Vogue of Triton",
                    "Title": "Ambient power",
                    "CompositeRating": "5.0",
                    "Voters": "4",
                    "TrackID": "10445"
                },
                {
                    "Artist": "Mister E",
                    "Title": "Dedic8ed 2 AsmoDeon",
                    "CompositeRating": "5.0",
                    "Voters": "4",
                    "TrackID": "13194"
                },
                {
                    "Artist": "Blackstar (Oddgeir
Hvidsten) and Time Traveller (Geir Rune Ladehaug)",
                    "Title": "Blue stars",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "10688"
                },
                {
                    "Artist": "Phobium",
                    "Title": "Tonik Funture
(phob)",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "14504"
                },
                {
                    "Artist": "4-mat",
                    "Title": "Eternity",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "13491"
                },
                {
                    "Artist": "Malmen",
                    "Title": "Edison Glam",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "14849"
                },
                {
                    "Artist": "Alexander Brandon",
                    "Title": "Menu song (Jazz
Jackrabbit Intro)",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "14114"
                },
                {
                    "Artist": "Riku Nuottajarvi",
                    "Title": "Hyperspace (From star
control)",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "11087"
                },
                {
                    "Artist": "Malmen of
Brainstorm",
                    "Title": "Devotion shuttle",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "13373"
                },
                {
```

```
                    "Artist": "MyVoice",
                    "Title": "Alertia",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "14915"
                },
                {
                    "Artist": "Danko and Jogeir
Liljedahl",
                    "Title": "1992",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "12380"
                },
                {
                    "Artist": "Blue orian",
                    "Title": "Spiral galaxy",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "14160"
                },
                {
                    "Artist": "Sudokan",
                    "Title": "Thoughts Per Second",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "13150"
                },
                {
                    "Artist": "Selecta Novel",
                    "Title": "Simon the Sorcerer -
Dragon's Winter Cave",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "14421"
                },
                {
                    "Artist": "Nicsure",
                    "Title": "Occ-san-geen",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "14422"
                },
                {
                    "Artist": "Arpegiator of
Chryseis",
                    "Title": "My wolf 2",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "10663"
                },
                {
                    "Artist": "Dr. Awesome",
                    "Title": "Session",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "10927"
                }
            ],
            "BottomTracks": [
                {
                    "Artist": "Sysfins",
                    "Title": "The aooga mooca
song",
                    "CompositeRating": "1.0",
                    "Voters": "3",
                    "TrackID": "15657"
                },
                {
                    "Artist": "Uctumi",
                    "Title": "Flimbo's quest
medley",
                    "CompositeRating": "1.0",
                    "Voters": "2",
                    "TrackID": "14045"
                },
                {
                    "Artist": "Audiomonster",
                    "Title": "Florence",
                    "CompositeRating": "1.0",
                    "Voters": "1",
                    "TrackID": "11963"
                },
                {
                    "Artist": "RawArgon",
                    "Title": "They drew 1st Blood
Orangeade",
                    "CompositeRating": "1.0",
                    "Voters": "1",
                    "TrackID": "14518"
                },
                {
                    "Artist": "Jam & Spoon
productions",
                    "Title": "Dansa uppa dekki",
                    "CompositeRating": "1.0",
                    "Voters": "1",
                    "TrackID": "12486"
                },
                {
                    "Artist": "Bst",
                    "Title": "Mayan Paradise*Dim
System",
                    "CompositeRating": "1.0",
                    "Voters": "1",
                    "TrackID": "14267"
                },
                {
                    "Artist": "mA2E / Desire",
                    "Title": "Overjoyed by groove",
                    "CompositeRating": "1.0",
                    "Voters": "1",
                    "TrackID": "11977"
                },
                {
                    "Artist": "Beldoroon",
```

```
                "Title": "Come allye (remix)",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "13778"
            },
            {
                "Artist": "Phaser",
                "Title": "The return of music",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "10996"
            },
            {
                "Artist": "Speechless",
                "Title": "Mistake",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "13284"
            },
            {
                "Artist": "Dr. Awesome",
                "Title": "Empty Spaces",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "10762"
            },
            {
                "Artist": "Psirius",
                "Title": "Musical cloister",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "14102"
            },
            {
                "Artist": "The Muzycant",
                "Title": "Guitar Slinger V2",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "14385"
            },
            {
                "Artist": "Liam the Lemming",
                "Title": "Auf wiedersehen Monty
theme",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "14156"
            },
            {
                "Artist": "Crk",
                "Title": "Beatex",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "10624"
            },
            {
                "Artist": "Mark M. Salud",
                "Title": "1-2-3-DanceMon",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "14953"
            },
            {
                "Artist": "Necros of FM",
                "Title": "The grey note",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "13936"
            },
            {
                "Artist": "Ceekayed",
                "Title": "Battle beyond the
dark",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "13946"
            },
            {
                "Artist": "K. Jose",
                "Title": "It's Time",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "13957"
            },
            {
                "Artist": "Madminder",
                "Title": "Shadow Angel",
                "CompositeRating": "1.0",
                "Voters": "1",
                "TrackID": "14478"
            }
        ],
        "TopArtists": [
            {
                "Artist": "Xyce",
                "CompositeRating": "5.0",
                "Voters": "19"
            },
            {
                "Artist": "Slash_Atd",
                "CompositeRating": "5.0",
                "Voters": "10"
            },
            {
                "Artist": "Arachno",
                "CompositeRating": "5.0",
                "Voters": "6"
            },
            {
                "Artist": "ArchAngel",
                "CompositeRating": "5.0",
                "Voters": "5"
            },
            {
                "Artist": "Maktone",
                "CompositeRating": "5.0",
```

```
            "Voters": "5"
        },
        {
            "Artist": "Vogue",
            "CompositeRating": "5.0",
            "Voters": "5"
        },
        {
            "Artist": "Nusrat",
            "CompositeRating": "5.0",
            "Voters": "4"
        },
        {
            "Artist": "Warder",
            "CompositeRating": "5.0",
            "Voters": "4"
        },
        {
            "Artist": "cs127",
            "CompositeRating": "5.0",
            "Voters": "4"
        },
        {
            "Artist": "AceMan",
            "CompositeRating": "5.0",
            "Voters": "4"
        },
        {
            "Artist": "Mister%20E",
            "CompositeRating": "5.0",
            "Voters": "4"
        },
        {
            "Artist": "Laamaa",
            "CompositeRating": "5.0",
            "Voters": "4"
        },
        {
            "Artist": "Riku%20Nuottajarvi",
            "CompositeRating": "5.0",
            "Voters": "3"
        },
        {
            "Artist": "Danko",
            "CompositeRating": "5.0",
            "Voters": "3"
        },
        {
            "Artist": "Saga%20Musix",
            "CompositeRating": "5.0",
            "Voters": "3"
        },
        {
            "Artist": "Arpegiator",
            "CompositeRating": "5.0",
            "Voters": "3"
        },
        {
            "Artist":
"Bacter%20vs%20Saga%20musix",
            "CompositeRating": "5.0",
            "Voters": "3"
        },
        {
            "Artist": "Phobium",
            "CompositeRating": "5.0",
            "Voters": "3"
        },
        {
            "Artist":
"Blackstar%20(Oddgeir%20Hvidsten)",
            "CompositeRating": "5.0",
            "Voters": "3"
        },
        {
            "Artist": "Dippy",
            "CompositeRating": "5.0",
            "Voters": "3"
        }
    ],
    "BottomArtists": [
        {
            "Artist": "Sysfins",
            "CompositeRating": "1.0",
            "Voters": "3"
        },
        {
            "Artist": "Crk",
            "CompositeRating": "1.0",
            "Voters": "1"
        },
        {
            "Artist": "Madminder",
            "CompositeRating": "1.0",
            "Voters": "1"
        },
        {
            "Artist": "Sam",
            "CompositeRating": "1.0",
            "Voters": "1"
        },
        {
            "Artist": "RawArgon",
            "CompositeRating": "1.0",
            "Voters": "1"
        },
        {
            "Artist": "Beldoroon",
            "CompositeRating": "1.0",
            "Voters": "1"
        },
        {
            "Artist": "Ceekayed",
            "CompositeRating": "1.0",
            "Voters": "1"
        },
```

```
                {
                    "Artist":
"Jam%20&%20Spoon%20productions",
                    "CompositeRating": "1.0",
                    "Voters": "1"
                },
                {
                    "Artist": "Speechless",
                    "CompositeRating": "1.0",
                    "Voters": "1"
                },
                {
                    "Artist": "mA2E%20/%20Desire",
                    "CompositeRating": "1.0",
                    "Voters": "1"
                },
                {
                    "Artist": "Mark%20M.%20Salud",
                    "CompositeRating": "1.0",
                    "Voters": "1"
                },
                {
                    "Artist": "NaBo",
                    "CompositeRating": "1.0",
                    "Voters": "1"
                },
                {
                    "Artist": "UsE",
                    "CompositeRating": "1.0",
                    "Voters": "1"
                },
                {
                    "Artist": "The%20Muzycant",
                    "CompositeRating": "1.0",
                    "Voters": "1"
                },
                {
                    "Artist": "Efenstor",
                    "CompositeRating": "2.0",
                    "Voters": "2"
                },
                {
                    "Artist": "Audiomonster",
                    "CompositeRating": "2.0",
                    "Voters": "2"
                },
                {
                    "Artist": "View",
                    "CompositeRating": "2.0",
                    "Voters": "1"
                },
                {
                    "Artist": "Tomi%20Blinnikka",
                    "CompositeRating": "2.0",
                    "Voters": "1"
                },
                {
                    "Artist": "Galaxy",
                    "CompositeRating": "2.0",
                    "Voters": "1"
                },
                {
                    "Artist": "Higher%20(than)",
                    "CompositeRating": "2.0",
                    "Voters": "1"
                }
            ],
            "StreamingCountries": [
                {
                    "Country": "Germany",
                    "Count": "930"
                },
                {
                    "Country": "Canada",
                    "Count": "590"
                },
                {
                    "Country": "United States",
                    "Count": "442"
                },
                {
                    "Country": "Croatia",
                    "Count": "332"
                },
                {
                    "Country": "Russia",
                    "Count": "314"
                }
            ],
            "StreamingAgents": [
                {
                    "Agent": "FreeAmp/2.x",
                    "Count": "559"
                },
                {
                    "Agent": "BASS/2.4",
                    "Count": "401"
                },
                {
                    "Agent": "Screamer Radio v0.4.4
beta (20100924)",
                    "Count": "327"
                }
            ],
            "LogoffPlay": [
                {
                    "Track": "Berluskani-Flimbo in
tyrol (Arok-remix).",
                    "NumberOfPlays": "8",
                    "StationID": "1"
                },
                {
                    "Track": "Yannis Brown-Banjo
Jamming.",
                    "NumberOfPlays": "7",
                    "StationID": "1"
```

```
                },
                {
                    "Track": "Trackerartist-
Ec#24.",
                    "NumberOfPlays": "7",
                    "StationID": "1"
                }
            ],
            "Duplicates": [
                {
                    "FullArtist": "4-mat",
                    "Title": "Eternity",
                    "TrackID": "13491"
                },
                {
                    "FullArtist": "4-mat",
                    "Title": "Eternity",
                    "TrackID": "15212"
                },
                {
                    "FullArtist": "Arpegiator of
Chryseis",
                    "Title": "My wolf 2",
                    "TrackID": "10663"
                },
                {
                    "FullArtist": "Arpegiator of
Chryseis",
                    "Title": "My wolf 2",
                    "TrackID": "11987"
                },
                {
                    "FullArtist": "C512w",
                    "Title": "Going to the moon",
                    "TrackID": "15273"
                }
            ]
        },
        {
            "StationID": "2",
            "StationName": "Best of ERICADE.radio",
            "TotalLength": "199.2",
            "TracksOnStation": "178",
            "Trackerstats": [
                {
                    "TrackerType": "Podcast
episode",
                    "Tracks": "169",
                    "Percent": "94.9",
                    "StationID": "2"
                },
                {
                    "TrackerType": "Live broadcast
rerun",
                    "Tracks": "8",
                    "Percent": "4.5",
                    "StationID": "2"
                },
                {
                    "TrackerType": "Amiga 4-channel
module",
                    "Tracks": "1",
                    "Percent": "0.6",
                    "StationID": "2"
                }
            ],
            "Requests": [],
            "TopTracks": [
                {
                    "Artist": "Flashback",
                    "Title": "144. A tracked
christmas",
                    "CompositeRating": "5.0",
                    "Voters": "4",
                    "TrackID": "15436"
                },
                {
                    "Artist": "Flashback",
                    "Title": "134. Xmas special",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "15011"
                },
                {
                    "Artist": "Flashback",
                    "Title": "92. As featured on
the radio",
                    "CompositeRating": "5.0",
                    "Voters": "3",
                    "TrackID": "12222"
                },
                {
                    "Artist": "Flashback",
                    "Title": "133. Xmas 2024",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "15009"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "16. Doing
miscellaneous stuff",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9675"
                },
                {
                    "Artist": "Flashback",
                    "Title": "53. The creation of a
station",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9682"
                },
                {
                    "Artist": "Flashback",
```

```
                    "Title": "70. Cautious
optimism",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9933"
                },
                {
                    "Artist": "The ERICADE Radio
Network",
                    "Title": "The \"One year
anniversary broadcast 2021\"  - third hour",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9696"
                },
                {
                    "Artist": "Flashback",
                    "Title": "72. 8!ghtBM on every
chart",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9981"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "11. Two shades of new
year. Hour 2/3.",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9733"
                },
                {
                    "Artist": "Flashback",
                    "Title": "143. Another mix
tape",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "15414"
                },
                {
                    "Artist": "ericade.radio",
                    "Title": "Its news time",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9589"
                },
                {
                    "Artist": "Flashback",
                    "Title": "115. The Spirit of
Retro and Xmas Part 1",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "14426"
                },
                {
                    "Artist": "Flashback",
                    "Title": "69. A new year -
2022, second hour",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9913"
                },
                {
                    "Artist": "Edison Party 2021",
                    "Title": "The Edison 2021 Live
Broadcast hour 3",
                    "CompositeRating": "5.0",
                    "Voters": "2",
                    "TrackID": "9665"
                },
                {
                    "Artist": "Flashback",
                    "Title": "91. Forndata Summer
2022",
                    "CompositeRating": "5.0",
                    "Voters": "1",
                    "TrackID": "12205"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "21. Look at you,
hacker!",
                    "CompositeRating": "5.0",
                    "Voters": "1",
                    "TrackID": "9672"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "15. Return of the
Tekmann",
                    "CompositeRating": "5.0",
                    "Voters": "1",
                    "TrackID": "9680"
                },
                {
                    "Artist": "Edison Party 2021",
                    "Title": "The Edison 2021 Live
Broadcast hour 5",
                    "CompositeRating": "5.0",
                    "Voters": "1",
                    "TrackID": "9681"
                },
                {
                    "Artist": "Flashback",
                    "Title": "44. The fly-over part
of summer",
                    "CompositeRating": "5.0",
                    "Voters": "1",
                    "TrackID": "9686"
                }
            ],
            "BottomTracks": [
                {
                    "Artist": "Flashback",
                    "Title": "147. A brand new
tracked 2026",
```

```
                    "CompositeRating": "1.0",
                    "Voters": "1",
                    "TrackID": "15490"
                },
                {
                    "Artist": "Flashback",
                    "Title": "108. Chillaxing while
waiting",
                    "CompositeRating": "1.0",
                    "Voters": "1",
                    "TrackID": "12961"
                },
                {
                    "Artist": "Flashback",
                    "Title": "135. Springtime",
                    "CompositeRating": "2.0",
                    "Voters": "2",
                    "TrackID": "15165"
                },
                {
                    "Artist": "Flashback",
                    "Title": "40. It feels good to
be bad",
                    "CompositeRating": "2.0",
                    "Voters": "1",
                    "TrackID": "9606"
                },
                {
                    "Artist": "Flashback",
                    "Title": "60. Among saints and
sinners",
                    "CompositeRating": "2.5",
                    "Voters": "4",
                    "TrackID": "9689"
                },
                {
                    "Artist": "Flashback",
                    "Title": "54. Failure is the
interesting option",
                    "CompositeRating": "2.5",
                    "Voters": "2",
                    "TrackID": "9634"
                },
                {
                    "Artist": "Flashback",
                    "Title": "89. In loving memory
of Vangelis",
                    "CompositeRating": "3.0",
                    "Voters": "2",
                    "TrackID": "12168"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "20. All you need is
love",
                    "CompositeRating": "3.0",
                    "Voters": "1",
                    "TrackID": "9615"
                },
                {
                    "Artist": "Flashback",
                    "Title": "69. A new year -
2022, first hour",
                    "CompositeRating": "3.0",
                    "Voters": "1",
                    "TrackID": "9910"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "33. Tracks from the
past",
                    "CompositeRating": "4.0",
                    "Voters": "3",
                    "TrackID": "9600"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "22. Another lazy
sunday evening",
                    "CompositeRating": "4.0",
                    "Voters": "2",
                    "TrackID": "9637"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "4. Demos and groups",
                    "CompositeRating": "4.0",
                    "Voters": "2",
                    "TrackID": "9641"
                },
                {
                    "Artist": "Flashback",
                    "Title": "69. A new year -
2022, third hour",
                    "CompositeRating": "4.0",
                    "Voters": "2",
                    "TrackID": "9911"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "27. A pox on all your
Amigas",
                    "CompositeRating": "4.0",
                    "Voters": "1",
                    "TrackID": "9669"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "18. From the desk of
miscellaneousity",
                    "CompositeRating": "4.0",
                    "Voters": "1",
                    "TrackID": "9676"
                },
                {
                    "Artist": "Flashback",
```

```
                    "Title": "46. Do you grok the
lingo?",
                    "CompositeRating": "4.0",
                    "Voters": "1",
                    "TrackID": "9687"
                },
                {
                    "Artist": "Flashback",
                    "Title": "130. June",
                    "CompositeRating": "4.0",
                    "Voters": "1",
                    "TrackID": "14830"
                },
                {
                    "Artist": "ericade.radio",
                    "Title": "This is the best of
Ericade.radio, here is the latest from Feature
Story News",
                    "CompositeRating": "4.0",
                    "Voters": "1",
                    "TrackID": "9804"
                },
                {
                    "Artist": "ericade.radio",
                    "Title": "You are listening to
the best of Ericade.radio, here is the news",
                    "CompositeRating": "4.0",
                    "Voters": "1",
                    "TrackID": "9806"
                },
                {
                    "Artist": "Amiga Flashback",
                    "Title": "17. Standing on the
brink in the eve of destruction",
                    "CompositeRating": "4.0",
                    "Voters": "1",
                    "TrackID": "9594"
                }
            ],
            "TopArtists": [
                {
                    "Artist":
"The%20ERICADE%20Radio%20Network",
                    "CompositeRating": "5.0",
                    "Voters": "2"
                },
                {
                    "Artist":
"Edison%20Party%202021",
                    "CompositeRating": "4.8",
                    "Voters": "6"
                },
                {
                    "Artist": "Flashback",
                    "CompositeRating": "4.6",
                    "Voters": "63"
                },
                {
                    "Artist": "ericade.radio",
                    "CompositeRating": "4.5",
                    "Voters": "4"
                },
                {
                    "Artist": "Amiga%20Flashback",
                    "CompositeRating": "4.4",
                    "Voters": "26"
                }
            ],
            "BottomArtists": [
                {
                    "Artist": "Amiga%20Flashback",
                    "CompositeRating": "4.4",
                    "Voters": "26"
                },
                {
                    "Artist": "ericade.radio",
                    "CompositeRating": "4.5",
                    "Voters": "4"
                },
                {
                    "Artist": "Flashback",
                    "CompositeRating": "4.6",
                    "Voters": "63"
                },
                {
                    "Artist":
"Edison%20Party%202021",
                    "CompositeRating": "4.8",
                    "Voters": "6"
                },
                {
                    "Artist":
"The%20ERICADE%20Radio%20Network",
                    "CompositeRating": "5.0",
                    "Voters": "2"
                }
            ],
            "StreamingCountries": [
                {
                    "Country": "Germany",
                    "Count": "930"
                },
                {
                    "Country": "Canada",
                    "Count": "590"
                }
            ],
            "StreamingAgents": [
                {
                    "Agent": "FreeAmp/2.x",
                    "Count": "559"
                },
                {
                    "Agent": "BASS/2.4",
                    "Count": "401"
                },
```

```
                {
                    "Agent": "Screamer Radio v0.4.4
beta (20100924)",
                    "Count": "327"
                }
            ],
            "LogoffPlay": [
                {
                    "Track": "Flashback-113. Just
listening to the radio.",
                    "NumberOfPlays": "14",
                    "StationID": "2"
                },
                {
                    "Track": "Amiga Flashback-21.
Look at you, hacker!.",
                    "NumberOfPlays": "11",
                    "StationID": "2"
                }
            ],
            "Duplicates": [
                {
                    "FullArtist": "4-mat",
                    "Title": "Eternity",
                    "TrackID": "13491"
                },
                {
                    "FullArtist": "4-mat",
                    "Title": "Eternity",
                    "TrackID": "15212"
                },
                {
                    "FullArtist": "Arpegiator of
Chryseis",
                    "Title": "My wolf 2",
                    "TrackID": "10663"
                }
            ]
        }
    ]
}
```

Statistics by FilterType

Returns a single category of statistics instead of the whole bundle. The response schema is identical to the Statistics endpoint, with only the requested section populated.

`POST` `https://api.ericade.net/radio/v2/stats/<filtertype>`

Replace `<filtertype>` in the URL with one of the values below.

Filter types

VALUE	POPULATES
`most-requested`	`Requests`
`stars`	`TopTracks`, `BottomTracks`, `TopArtists`, `BottomArtists`
`tracktypes`	`Trackerstats`
`countries`	`StreamingCountries`
`agents`	`StreamingAgents`
`logoffplay`	`LogoffPlay`
`duplicates`	`Duplicates`
`listeners`	Accepted by the router, but no section is currently keyed to it — the response comes back with the station header only.

> Anything outside this list is rejected with HTTP 400 and the message *"Unknown stats type. Permitted types are: most-requested, stars, listeners, tracktypes, countries, agents, logoffplay, duplicates."* A `latest-tracks` type is planned but is not currently accepted — earlier revisions of this document listed it as a valid value.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	(Not implemented) Accepted, but the response always covers every station.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	Number of rows in each top/bottom list. Defaults to 10; must be 1–100.	10
`Days`	No	How far back the rating and request windows reach, in days. Defaults to 30.	7

> Unlike the unfiltered Statistics endpoint, this one *does* forward `Limit` and `Days`. If you need to tune either, use this path with the filter type you want.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "StationID": 1,
  "Limit": 20,
  "Days": 7
}
```

Listening

Returns a live snapshot of who is listening right now: how many listeners there are, which countries they are listening from, what kind of connection they use and how long the longest current session has lasted. It also returns how many listeners the station has had so far today.

`POST` `https://api.ericade.net/radio/v2/listening`

The live figures are read from the streaming server on every call; nothing is cached. Only counts are returned — no IP addresses, ISPs or locations below country level.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	No	The station to report on. Omit it to get the combined figures for every station.	1 = ericade.radio 2 = Best of ericade.radio

How listeners are counted

- `RawListeners` is every client connected to the streaming server, exactly as the server reports it.
- Every other figure is *filtered*: clients on the station's own network and other excluded addresses are left out, and so are known bots and directory checkers (stream checkers, radio-directory crawlers), matched on their exact user agent. `Listeners` is therefore never higher than `RawListeners`.
- A **modern** client is connected through the station's stream proxy. A **legacy** client is connected directly to the streaming server. Every filtered listener is one or the other, so `LegacyListeners` + `ModernListeners` = `Listeners`.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 1
}
```

Response

Unlike most of the older endpoints, every count here is a JSON number, not a string.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Listeners`	Listeners connected right now, after filtering.	5
`RawListeners`	Clients connected right now as reported by the streaming server, before any filtering.	7
`Countries`	Object that maps each country to its number of filtered listeners, highest count first and ties in alphabetical order. Listeners whose country is not known are grouped under `Unknown`. An empty object `{}` when nobody is listening.	{"Sweden": 3, "Norway": 1, "Unknown": 1}

ATTRIBUTE	DESCRIPTION	EXAMPLE
`LegacyListeners`	Filtered listeners connected directly to the streaming server.	1
`ModernListeners`	Filtered listeners connected through the stream proxy.	4
`LongestListening`	How long the longest-connected current listener has been listening, as `hh:mm:ss`. Hours do not wrap at 24, so a listener connected for 27 hours shows `27:05:10`. `00:00:00` when nobody is listening.	03:12:45
`TotalListenersToday`	Unique listeners since midnight (server time) who have listened for more than ten minutes. Counts both finished sessions and current listeners who have already passed ten minutes.	23

Example response

```
{
  "Listeners": 5,
  "RawListeners": 7,
  "Countries": {
    "Sweden": 3,
    "Norway": 1,
    "Unknown": 1
  },
  "LegacyListeners": 1,
  "ModernListeners": 4,
  "LongestListening": "03:12:45",
  "TotalListenersToday": 23
}
```

About `TotalListenersToday`. Listeners are told apart by IP address. Someone who reconnects several times, or listens to both stations, is counted once, and several people sharing one address (a household or an office) also count as one. A finished session counts on the day it ended. Ten minutes is the station's minimum streaming time, the same threshold the Statistics endpoint uses.

Many JSON parsers keep the order of `Countries` as sent, but JSON objects are unordered by definition. If the order matters to you, sort on the client.

Errors

Failures use the `message` / `subcode` / `submessage` envelope with HTTP 400:

CAUSE	`SUBMESSAGE`
Missing or invalid token	User token invalid or empty.
`StationID` is not a number	StationID must be a number.
Unknown `StationID`	Station configuration is not valid or the station does not exist.
The streaming server could not be reached, or returned something unreadable	Could not retrieve listener data from the streaming server. Please try again.

CAUSE	`SUBMESSAGE`
Database or other server-side error	A temporary error occurred while retrieving listening statistics. Please try again.

If the streaming server cannot be read for any requested station, the call fails instead of reporting zero listeners. If the request runs out of time or memory on the server, it returns HTTP 500 with `subcode` `Fatal error` and the same “temporary error” `submessage`.

Per-listener detail (IP address, ISP, city) is available only from the internal, IP-restricted `/radio/getlistening/` endpoint. See the v1 documentation.

Requests

Returns the current request queue status and recent request history. Useful for displaying a live request line on a website.

`POST` `https://api.ericade.net/radio/v2/requests`

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint.
`StationID`	Yes	The station to get request data from.	1 = ericade.radio 2 = Best of ericade.radio
`Limit`	No	(Not implemented) Limit the number of requests returned.	

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "StationID": 1,
  "Limit": 10
}
```

Response

ATTRIBUTE	DESCRIPTION	EXAMPLE
`RequestStatistics`	Array containing one object with queue wait-time statistics.	Array[]
`RequestStatistics ⇒ TracksInQueue`	Number of tracks currently waiting in the request queue.	3
`RequestStatistics ⇒ YourQueueNumber`	Position a new request would occupy in the queue.	4
`RequestStatistics ⇒ ExpectedWaitingTimeInMinutes`	Estimated wait in minutes until a new request would be played.	48
`RequestStatistics ⇒ ExpectedWaitingTime`	Estimated wait expressed as a human-readable string, suitable for display on a website.	48 minutes
`CurrentRequests`	Array of the five most recent requests (both waiting and already played).	Array[]
`CurrentRequests ⇒ Song`	Requested track expressed as “<artist> – <title>”.	Tommy Fjelldal - Edificated dreams
`CurrentRequests ⇒ Requested`	Time the request was submitted, as a Unix epoch string.	1774902144
`CurrentRequests ⇒ TrackID`	Unique track identifier string.	15715

ATTRIBUTE	DESCRIPTION	EXAMPLE
`CurrentRequests ⇒ Source`	Origin of the request.	Internet or Discord
`CurrentRequests ⇒ State`	Whether the request is still waiting or has already been played.	Waiting or Played

Example response


```
{
  "RequestStatistics": [
    {
      "TracksInQueue": "3",
      "YourQueueNumber": "4",
      "ExpectedWaitingTimeInMinutes": "48",
      "ExpectedWaitingTime": "48 minutes"
    }
  ],
  "CurrentRequests": [
    {
      "Song": "Tommy Fjelldal - Edificated dreams",
      "Requested": "1774902144",
      "TrackID": "15715",
      "Source": "Internet",
      "State": "Waiting"
    },
    {
      "Song": "Arpegiator of Chryseis - My wolf 2",
      "Requested": "1774902734",
      "TrackID": "10663",
      "Source": "Internet",
      "State": "Waiting"
    },
    {
      "Song": "Dr Future - Plastic pop (Plastic kills life mix)",
      "Requested": "1774903007",
      "TrackID": "14922",
      "Source": "Internet",
      "State": "Waiting"
    },
    {
      "Song": "X-Ceed of Sco - Let's go for a ride",
      "Requested": "1774902096",
      "TrackID": "15711",
      "Source": "Internet",
      "State": "Played"
    },
    {
      "Song": "Alexander Brandon - Frolick lane",
      "Requested": "1774902091",
      "TrackID": "15714",
      "Source": "Internet",
      "State": "Played"
    }
  ]
}
```

Add Request

Submits a listener request to the station's request queue.

`POST` `https://api.ericade.net/radio/v2/addrequest`

> This endpoint uses `Password` instead of `Token` as the field name for the JWT value (legacy naming), and its response uses the capitalised `Success` / `Message` envelope rather than the lower-case one used elsewhere.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Password`	Yes	JWT token that authenticates the call.	Get from the get-token endpoint. Note: field is named `Password`, not `Token`.
`StationID`	Yes	The station to submit the request to. The station must have requests enabled.	1 = ericade.radio 2 = Best of ericade.radio
`TrackID`	Yes	Unique identifier of the track to request. Obtain from the Search or Now Playing endpoints.	10663
`Source`	Yes	Origin of the request. Can control which jingle plays before the track. Max 20 characters.	Internet or Discord
`Requester`	No	The self-selected display name of the requester. Max 120 characters.	Caller #9
`Greeting`	No	*(Deprecated) A user-selectable greeting to the show host. Max 250 characters.*	Hi! Love your music!

Workflow

1. Use Search to find the track and obtain its `TrackID`.
2. Check that `TrackCanBeRequested` is 1; if not, show `RequestVerdict` to the user.
3. Post to this endpoint with the `TrackID`.

Limits

The request is rejected, with an explanatory `Message`, when any of the following applies:

- The station does not allow requests.
- The queue has already reached the station's maximum pending requests.
- The caller has submitted 5 or more requests in the last 5 hours.
- The track or its artist was played too recently — the reason is echoed from the eligibility check.

Example request

```
{
  "Password": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 1,
  "Requester": "Listener",
  "Source": "Internet",
  "TrackID": 10663
}
```

Response

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Success`	Indicates whether the request was accepted. Either `Success` or `Failure`.	Success
`Message`	Human-facing message suitable for display to the end user, including the expected waiting time on success and the reason on failure.	Success! The tune you requested has been added to the playlist. Expected waiting time until play: 48 minutes.

Example response

```
{
  "Success": "Success",
  "Message": "Success! The tune you requested has been added to the playlist. Expected waiting time until play: 48 minutes."
}
```

Legacy route. The v1 endpoint `https://api.ericade.net/radio/addrequest/` still works and takes the same fields, but returns the lower-case `message` / `subcode` / `submessage` envelope instead. New integrations should use the v2 route above. Do not call it as `/radio/addrequest/index.php` — paths ending in `.php` under `/radio/` are 301-redirected to an extension-less path that does not resolve.

A `song-request` event fires on the Station Data WebSocket the moment the request is accepted. The greeting and the requester's IP are never broadcast.

Star Rating

Records a listener's 1–5 star rating for a track, then recalculates both the track's and the artist's composite ratings.

`POST` `https://api.ericade.net/radio/v2/rating`

This endpoint returns a third envelope shape: `success` is a real JSON boolean and the message key is lower-case `message`.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	JWT token that authenticates the call. The browser hash inside the token identifies the voter.	Get from the get-token endpoint.
`TrackID`	Yes	Unique identifier of the track being rated.	10663
`TrackRating`	Yes	The rating, 1–5. Single digit; anything outside the range is rejected.	4
`StationID`	No	Accepted, but ignored — the station is resolved from the track itself.	

Limits

- Maximum 5 ratings per minute per voter.
- The same voter may not rate the same track again within 3 days.
- Check `TrackCanBeStarred` on the track record before offering the control.

Latency is deliberate. Each call sleeps for 1–3 seconds before doing its work, as anti-spam throttling. Do not treat this as a timeout and do not retry on slow responses — a retry counts against the 5-per-minute limit.

Example request

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 ... ",
  "TrackID": 10663,
  "TrackRating": 4
}
```

Response

ATTRIBUTE	DESCRIPTION	EXAMPLE
`success`	Boolean. `true` when the rating was recorded.	true
`message`	Human-facing message suitable for display to the end user.	Stars updated.

Typical failure messages are `You may only star 5 items per minute.`, `You may not vote for this track again at this time` and `TrackRating must be between 1 – 5.`

Example response

```
{
  "success": true,
  "message": "Stars updated."
}
```

> A `song-rating` event fires on the Station Data WebSocket once the rating is recorded. The voter's IP and browser hash are never broadcast.

Station Override

Switches a station between normal broadcast and a manual override. With the override on, every now-playing response for the station returns the supplied artist and title instead of the track actually playing — used to announce a live show or display a special message.

`POST` `https://api.ericade.net/radio/v2/settings`

Operator endpoint. A normal listener token is accepted by the transport, but a successful call changes what every listener of the station sees. It also uses `Password` for the JWT and the capitalised `Success` / `Message` envelope.

Request

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Password`	Yes	JWT token that authenticates the call.	Note the field name — not `Token`.
`StationID`	Yes	The station to control. A settings row must already exist for it.	1 = ericade.radio 2 = Best of ericade.radio
`Override`	Yes	1 = show the supplied artist/title instead of the playing track. 0 = return to normal broadcast. Any other value is rejected.	0 or 1
`Artist`	No	Artist text to display while the override is active. Max 250 characters.	ericade.radio
`Title`	No	Title text to display while the override is active. Max 250 characters.	Live from Revision 2026

While the override is active, Now Playing returns `isOverride: 1` and Next Up returns the synthetic “Back to normal broadcast” entry. The `isLive` and `isRemote` flags on the track record are read from the settings row and are not settable through this endpoint.

Example request

```
{
  "Password": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "StationID": 1,
  "Override": 1,
  "Artist": "ericade.radio",
  "Title": "Live from Revision 2026"
}
```

Response

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Success`	Either `Success` or `Failure`.	Success

ATTRIBUTE	DESCRIPTION	EXAMPLE
`Message`	Human-facing message. `Override updated.` on success.	Override updated.

Example response

```
{
  "Success": "Success",
  "Message": "Override updated."
}
```

> When the override state actually changes, `now-playing` and `override` events fire on the Station Data WebSocket. Repeat calls that do not change the state fire nothing.

User Accounts: Overview

Four endpoints let people log in to radio.ericade.net: authenticate (login and user tokens), account (signup, e-mail confirmation, password reset, SceneID), profile (who somebody is, and their picture) and privileges (what they may do). Each takes an *action* as the last part of its path, e.g. `/radio/v2/account/create`.

> **Built for the website's PHP, not for browsers.** radio.ericade.net calls these server-to-server and keeps the user token in an `HttpOnly` cookie, where no script can reach it. They are reachable from anywhere, and everything below holds for any caller — but see *CORS* under How these differ.

Two tokens

FIELD	WHAT IT IS	NEEDED
`Token`	The ordinary API ticket every v2 endpoint takes — see Authentication. It identifies a browser. It proves nothing about a person.	On every call.
`UserToken`	Issued by authenticate/login. A JWE (RFC 7516, compact serialization, `alg: dir`, `enc: A256GCM`): five dot-separated parts, the second one empty. It is encrypted with a key only the API holds, so a client can store it and send it back but can neither read nor alter it. Behind every token is a server-side session; the token is honoured only while that session is alive.	On every call that acts as a logged-in user.

> Who the caller is comes from `UserToken` and from nothing else. `AccountID` in a request only selects *which* account an **administrator** is working on; sent by anybody else it is refused with HTTP 403. A caller's privilege is read from the database on every call, never from the token — so a promotion, a ban or a logout takes effect on the very next request.

Privileges

Every account holds exactly one, or none:

PRIVILEGE	MEANING	CAN LOG IN?
`administrator`	Highest. Manages accounts, profiles and privileges of others.	Yes
`contributor`	May write articles.	Yes
`user`	Regular user, accepted by an administrator.	Yes
`new`	Account and profile created, not yet accepted. May edit its own account and profile.	Yes
`banned`	No privileges, pending further investigation.	No — and every session is revoked the moment the ban is set.
`none`	Not a value that can be set: what is reported for an account whose privilege has been deleted.	No

They rank `administrator` > `contributor` > `user` > `new`: whoever holds one satisfies a requirement for any below it. `banned` and `none` satisfy nothing.

How these differ from the other v2 endpoints

- **One envelope, always.** Success and failure alike answer `{"message": "Success"|"Failure", "subcode": "…", "submessage": "…", …data}`. `subcode` is a short fixed string meant for code; `submessage` is a finished sentence meant for the person at the screen. It is plain text — escape it before printing it in HTML.
- **The status code is reliable** here, unlike on the data endpoints: 200 means it worked, anything else means it did not.
- **CORS:** `Access-Control-Allow-Origin` is `https://radio.ericade.net`, not `*`. These calls carry passwords and tokens; a script on somebody else's page has no business reading the answers. `OPTIONS` answers 204, any other method than `POST` answers 405.
- **Never cached:** every response is `Cache-Control: no-store`. The content type is the correctly spelled `application/json; charset=UTF-8`.
- **Text is stored as typed** and returned as typed: names, handles, cities and notes come back as raw UTF-8, not HTML-encoded as the track data is. Escape on output.

CODE	MEANING
`200`	Done.
`400`	A field is missing or invalid, or the action is unknown. `submessage` says which and why.
`401`	`Token` invalid (`subcode: Invalid token`), login failed (`Login failed`), or `UserToken` missing, expired, revoked or tampered with (`Not logged in` — always the same answer, whatever was wrong with it).
`403`	Logged in, but not allowed: too low a privilege, another person's `AccountID`, a wrong `CurrentPassword`, a suspended account, an unconfirmed e-mail address.
`404`	The account, profile, image or privilege does not exist.
`409`	Conflict: the user name is taken, a profile or privilege already exists, this is the last administrator.
`429`	Too many failed logins, signups or mail requests — see below. Carries `Retry-After`. Distinct from the API-wide rate limit, which also applies.
`500`	Temporary error; details are in the server's error log, never in the response.
`503`	`subcode: Not configured` — the server has no `$UserTokenKey` yet, so nobody can log in. See Setup.

Throttling and lockout

WHAT	LIMIT (DEFAULT)	THEN
Failed logins against one user name or e-mail address	5 in 15 minutes	429 for that name until the window passes, even with the right password and from any address. A successful login resets the count.
Failed logins from one address	20 in 15 minutes	429 for that address. Catches password spraying across many names.
Signups from one address	5 an hour	429
Reset / resend-confirmation requests from one address	10 an hour	429
Mails of one kind to one account	3 an hour	No more are sent. The caller is not told.

> **`ClientIP` and `UserAgent`.** A server calling on behalf of a visitor should pass the visitor's address and browser in these two fields, or every visitor appears to come from that server and one person guessing passwords locks out everybody. `ClientIP` is believed *only* when the request itself arrives from an address in `$UserTrustedFrontendIPRanges`; from anywhere else it is silently ignored and the connection's own address is used.

The User object

Returned as `User` by login, validate and refresh, and as `Account` by account/get and account/modify. It never contains the password hash.

ATTRIBUTE	DESCRIPTION	EXAMPLE
`AccountID`	Numeric id of the account.	42
`Username`	Unique login name.	daemon
`DisplayName`	What to call this person on a page: the scene handle, else first and last name, else the user name.	DJ Daemon
`Email`	Login and recovery address.	daemon@example.org
`PendingEmail`	A requested new address that has not been confirmed yet, or empty.	
`EmailVerified`	1 once the address has been confirmed.	1
`Privilege`	One of the privileges, or `none`.	user
`HasPassword`	0 for an account created through SceneID that has not set a password.	1
`SceneIDLinked`, `SceneIDDisplayName`	Whether the account is federated with SceneID, and as whom.	1, "Daemon ^ Ericade"
`FirstName`, `LastName`, `SceneHandle`	From the profile. Empty strings when not set.	
`ProfileImage`, `ProfileImageURL`	File name and full URL of the profile image, or empty.	https://radio.ericade.net/images/profiles/3f0c…a91.webp
`Created`, `LastLogin`	Unix epochs. `LastLogin` is 0 before the first login.	1789646400

The usual sequence

1. account/create — the account (privilege `new`) and its profile are created together; a confirmation link is mailed.
2. account/verify with the token from that link.
3. authenticate/login → `UserToken`. Store it where scripts cannot read it.
4. authenticate/validate on later requests: is the token still good, whose is it, what may they do?
5. An administrator accepts the account: privileges/modify `new` → `user`.

Authenticate

Creates, checks, renews and revokes user tokens.

`POST` `https://api.ericade.net/radio/v2/authenticate/<action>`

Actions: `login`, `validate`, `refresh`, `logout`. All take `Token`; `ClientIP` and `UserAgent` are optional on all (see above).

login

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	API ticket.	Get from the get-token endpoint.
`Identifier`	Yes	User name or e-mail address. Anything containing `@` is looked up as an address; user names cannot contain one.	Case-insensitive.
`Password`	Yes	The password, as typed. Max 1024 bytes.	
`Remember`	No	1 = the login lasts 30 days instead of 12 hours.	0 or 1

An unknown name, a wrong password and an account without a password all give the same 401 `Login failed`, after the same amount of work. Only a caller who has proven the password is told more: 403 `Suspended` (banned, or no privilege) or 403 `Email not verified` (offer account/resendverify).

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "Identifier": "daemon",
  "Password": "correct horse battery staple",
  "Remember": 1,
  "ClientIP": "203.0.113.10",
  "UserAgent": "Mozilla/5.0  … "
}
```

Response (login and refresh)

ATTRIBUTE	DESCRIPTION	EXAMPLE
`UserToken`	The JWE. Around 450 characters of `A-Z a-z 0-9 . _ -`.	eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0..Zk3…
`Expires`	Unix epoch at which the login ends.	1789689600
`User`	The User object.	Object{}

```
{
  "message": "Success",
  "subcode": "Logged in",
  "submessage": "Welcome back!",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0..Zk3sR0m1Qw8xT2pL.9vK … ",
  "Expires": 1789689600,
```

```
  "User": {
    "AccountID": 42,
    "Username": "daemon",
    "DisplayName": "DJ Daemon",
    "Email": "daemon@example.org",
    "PendingEmail": "",
    "EmailVerified": 1,
    "Privilege": "user",
    "HasPassword": 1,
    "SceneIDLinked": 0,
    "SceneIDDisplayName": "",
    "FirstName": "Erik",
    "LastName": "Zalitis",
    "SceneHandle": "DJ Daemon",
    "ProfileImage": "3f0c5a7e9b1d4c6f8a2b4d6e8f0a1c91.webp",
    "ProfileImageURL": "https://radio.ericade.net/images/profiles/3f0c5a7e9b1d4c6f8a2b4d6e8f0a1c91.webp",
    "Created": 1789646400,
    "LastLogin": 1789646460
  }
}
```


validate

Is this token good, and whose is it? This is the call an administrative page makes before it shows anything.

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	API ticket.	
`UserToken`	Yes	The token to check.	
`RequiredPrivilege`	No	When given, the response also says whether the user holds at least this.	administrator, contributor, user or new

Answers 200 with `Expires` and `User`; with `RequiredPrivilege` also `Authorized` (1 or 0). A valid token whose user is *not* authorized is still a 200 — check `Authorized`. A bad token is a 401.


```
{
  "message": "Success",
  "subcode": "Valid",
  "submessage": "The user token is valid.",
  "Expires": 1789689600,
  "User": { "AccountID": 42, "Username": "daemon", "Privilege": "user", "…": "…" },
  "RequiredPrivilege": "administrator",
  "Authorized": 0
}
```


refresh

Takes `Token` and `UserToken`; answers like login, with a new token of the same lifetime class (12 hours or 30 days). The old token keeps working for **60 more seconds** and then dies — long enough for a second browser tab that sent the old cookie at the same moment.

logout

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	API ticket.	
`UserToken`	Yes	The token to revoke.	
`AllSessions`	No	1 = revoke every session of the account, on every device.	0 or 1

Always answers 200 `Logged out` — a token that was already dead is as logged out as it gets.

Account

Adds, changes and deletes accounts; confirms e-mail addresses; resets lost passwords; federates with SceneID.

`POST` `https://api.ericade.net/radio/v2/account/<action>`

ACTION	WHO	DOES
`create`	Anyone	Creates the account, its profile and the privilege `new`; mails a confirmation link.
`verify`	Anyone with the mailed token	Confirms an e-mail address.
`resendverify`	Anyone	Mails a fresh confirmation link.
`resetrequest`	Anyone	Mails a link for choosing a new password.
`resetconfirm`	Anyone with the mailed token	Sets the new password.
`get`	Logged in	Returns the account.
`list`	Administrator	Lists accounts, newest first.
`modify`	Logged in	Changes user name, e-mail address or password.
`delete`	Logged in	Deletes the account and everything belonging to it.
`sceneidurl`, `sceneidlogin`, `sceneidlink`, `sceneidunlink`	See SceneID	

create

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Token`	Yes	API ticket.	
`Username`	Yes	3–40 characters: `A-Z a-z 0-9 . _ -`, starting with a letter or digit. Unique, case-insensitively. A handful of names (admin, root, support, …) are reserved.	daemon
`Email`	Yes	Stored lower-case. Unique.	
`Password`	Yes	10–128 characters. No composition rules; refused when too repetitive, built on a very common word, or containing the user name or the address. Stored as an Argon2id hash (bcrypt on a PHP built without Argon2).	
`SceneHandle`, `FirstName`, `LastName`	No	Seed the profile. Max 100 characters each; `<` and `>` are refused.	

The answer does not say whether the address was already registered. A signup with a known address gets the same 200 as a new one and creates nothing; the owner of the address is mailed instead ("somebody tried to register with your address"). A taken *user name* is reported (409 `Username taken`) — that cannot be kept secret from somebody choosing one. No `AccountID` is returned.


```
{
  "message": "Success",
  "subcode": "Account created",
  "submessage": "Almost there! We have sent a mail to daemon@example.org. Open the link in it to confirm
your address, then log in."
}
```


verify, resendverify

The mailed link is `https://radio.ericade.net/signup/?verify=<token>`; `verify` takes that token as `VerifyToken` (64 hex characters). It works once, for two days, and confirms exactly the address it was mailed to. A newer mail invalidates the older link. Only the SHA-256 of a token is ever stored.

`resendverify` takes `Identifier` (user name or address) and always answers the same 200, whether or not a mail was sent.

resetrequest, resetconfirm

`resetrequest` takes `Identifier` and always answers the same 200 `Request received`. A link (`https://radio.ericade.net/lost/?reset=<token>`, valid one hour, once) is mailed only to a **confirmed** address; an unconfirmed account is sent its confirmation mail again instead.

`resetconfirm` takes `ResetToken` and `NewPassword`. The password is checked *before* the token is spent, so a rejected password (400 `Invalid Password`) does not cost the link; a spent, expired or made-up token is 400 `Invalid link`. On success every session of the account is revoked and the owner is notified by mail.

get

Takes `UserToken`; an administrator may add `AccountID`. Returns `Account`, a User object.

list

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`UserToken`	Yes	An administrator's.	
`Search`	No	Matches the *start* of the user name, e-mail address or scene handle. `%` and `_` are literal.	Max 100 characters.
`Privilege`	No	Only accounts holding this.	A privilege, or `none`.
`Skip`, `Limit`	No	Pagination. `Limit` 1–100, default 25.	

Returns `Total`, `Skip`, `Limit` and `Accounts[]`, each with `AccountID`, `Username`, `Email`, `EmailVerified`, `Privilege`, `SceneHandle`, `FirstName`, `LastName`, `ProfileImageURL`, `SceneIDLinked`, `Created`, `LastLogin`.

modify

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`UserToken`	Yes		
`CurrentPassword`	On your own account	Changing your own credentials takes your current password — a session left open on a shared computer must not be enough to take the account over. 403 `Wrong password` otherwise.	Not asked of an account that has none (SceneID).
`Username`	No	New user name. 409 when taken.	
`Email`	No	New address. On your own account it is **parked** in `PendingEmail` and a confirmation link is mailed to it; the old address keeps working — for login and for password resets — until the new one is confirmed, so a typo cannot lock anybody out. The old address is told about the request.	
`NewPassword`	No	Same rules as at signup. Revokes every *other* session; the calling one survives.	Own account only. An administrator sending it for somebody else gets 403 — use `resetrequest`.
`AccountID`	No	Administrators: the account to change. No `CurrentPassword` needed; `Email` takes effect at once.	
`EmailVerified`	No	Administrators, on other accounts: set or clear the confirmed flag.	0 or 1

Returns the updated `Account`. 400 `Nothing to change` when no field differed.

delete

Takes `UserToken` and, on your own account, `CurrentPassword`; an administrator may name another `AccountID`. Removes the account, its profile, groups, links, privilege, sessions and mail tokens, and the profile image file. The audit log keeps its entries. 409 `Last administrator` when it would leave the site without one.

SceneID

SceneID is OAuth 2.0 (authorization code grant). The website owns the browser redirects and the `state` value; the API owns the client secret and everything done with the code — the secret never leaves the API. Off (and `sceneidurl` answers `Enabled: 0`) until `$SceneIDClientID` and `$SceneIDClientSecret` are set.

ACTION	TAKES	DOES
`sceneidurl`	`State`: 16–128 characters of `A-Z a-z 0-9 _ -`, random, kept in the visitor's session	Returns `Enabled` and `AuthorizeURL` — where to send the browser. Scope `basic user:email`.

ACTION	TAKES	DOES
`sceneidlogin`	`Code` from the callback; `Remember`	Exchanges the code, asks SceneID who it is, and logs in the account federated with that SceneID user — answering like login. The first time, an account is created: user name from the SceneID display name, privilege `new`, no password, address unconfirmed (a confirmation link is mailed; until it is used no reset link will go to that address).
`sceneidlink`	`UserToken`, `Code`	Connects the logged-in account to the SceneID user. 409 when that SceneID is connected elsewhere.
`sceneidunlink`	`UserToken`, `CurrentPassword`	Disconnects. 409 `No password` when SceneID is the only way into the account.

An existing account is never adopted on the strength of a matching e-mail address. If SceneID reports an address that already belongs to an account here, `sceneidlogin` answers 409 `Account exists` and links nothing: otherwise whoever controls that address at SceneID would own the account. The owner logs in with the password and uses `sceneidlink`. — The callback must reject a `state` that is not the one in the visitor's own session, or an attacker can hand a victim a link carrying the attacker's code.

Profile

What somebody says about themselves. One per account, created together with it.

`POST` `https://api.ericade.net/radio/v2/profile/<action>`

Actions: `add`, `get`, `modify`, `delete`, `image/add`, `image/modify`, `image/delete`. All take `Token` and `UserToken` and act on the caller's own profile; an administrator may add `AccountID` to act on another. All but `delete` return the full `Profile`.

Fields (add, modify)

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`FirstName`, `LastName`, `SceneHandle`, `City`, `Country`	No	Plain text, max 100 characters. Any printable UTF-8 except `<` and `>`; control characters and Unicode direction overrides are refused; runs of whitespace collapse to one space.	Scene handles are not unique.
`Email`	No	Contact address shown on the profile. Seeded from the account's, but independent of it — changing it changes nothing about login.	
`Phone`	No	5–40 characters of digits, spaces and `+ ( ) . / -`.	+46 8 123 45 67
`Groups`	No	Array of demo group names, max 20, each as the text fields above. Order is kept; blanks and case-insensitive duplicates are dropped.	["Fairlight", "Razor 1911"]
`Links`	No	Array of `{"Type", "URL"}`, max 15. `URL` must be a complete `http://` or `https://` address, max 500 characters, with no spaces, quotes, angle brackets or backslashes — `javascript:`, `data:` and scheme-less addresses are refused.	`Type`: pouet, demozoo, csdb, modarchive, bandcamp, soundcloud, youtube, github, mastodon, bluesky, website, other

modify changes what it is sent. A field that is absent is left alone; one sent as `""` is cleared. `Groups` and `Links`, when present, *replace* the whole array — send `[]` to empty one. `add` is only for an account whose profile has been deleted (409 `Profile exists` otherwise); `delete` removes the profile, its arrays and its image, and leaves the account.


```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0 .. Zk3 … ",
  "SceneHandle": "DJ Daemon",
  "City": "Stockholm",
  "Country": "Sweden",
  "Phone": "",
  "Groups": ["Ericade"],
  "Links": [
    { "Type": "demozoo", "URL": "https://demozoo.org/sceners/12345/" },
    { "Type": "pouet", "URL": "https://www.pouet.net/user.php?who=12345" }
  ]
}
```

```
{
  "message": "Success",
  "subcode": "Profile modified",
  "submessage": "The profile has been saved.",
  "Profile": {
    "AccountID": 42,
    "FirstName": "Erik",
    "LastName": "Zalitis",
    "SceneHandle": "DJ Daemon",
    "ProfileImage": "3f0c5a7e9b1d4c6f8a2b4d6e8f0a1c91.webp",
    "ProfileImageURL": "https://radio.ericade.net/images/profiles/3f0c5a7e9b1d4c6f8a2b4d6e8f0a1c91.webp",
    "Groups": ["Ericade"],
    "Email": "daemon@example.org",
    "Phone": "",
    "City": "Stockholm",
    "Country": "Sweden",
    "Links": [
      { "Type": "demozoo", "URL": "https://demozoo.org/sceners/12345/" },
      { "Type": "pouet", "URL": "https://www.pouet.net/user.php?who=12345" }
    ]
  }
}
```


Profile images

ACTION	TAKES	DOES
`image/add`	`FileData`	Sets the image of a profile that has none. 409 `Image exists` otherwise.
`image/modify`	`FileData`	Replaces the image and removes the old file. 404 `No image` when there is none to replace.
`image/delete`		Removes the image and its file.

`FileData` is the image file, base64-encoded (plain base64, no `data:` prefix): JPEG, PNG, GIF or WebP, at most 2 MB, between 32×32 and 5000×5000 pixels and at most 12.5 megapixels.

> **What was uploaded is never what is stored.** The dimensions are read from the header and checked *before* the image is decoded (a few kilobytes can claim to be 50000×50000). The image is then decoded, turned upright (EXIF), centre-cropped to a square, scaled down — never up — to 512×512, and re-encoded as WebP under a random 32-hex-character name in `/images/profiles/`. Nothing of the original bytes, its metadata or its file name survives, and no part of the request can influence the path. SVG is refused.

Privileges

What an account may do — see the list of privileges.

`POST` `https://api.ericade.net/radio/v2/privileges/<action>`

ACTION	WHO	TAKES	DOES
`get`	Logged in	`UserToken`; administrators may add `AccountID`	Returns `AccountID`, `Username`, `Privilege`, `Granted` (epoch), `GrantedBy` (AccountID, 0 = the system) and `Note`.
`list`	Administrator	`UserToken`; optional `Privilege`, `Skip`, `Limit`	Returns `Counts` (how many accounts hold each privilege) and `Privileges[]`. Filter on `new` for the queue of accounts waiting to be accepted.
`add`	Administrator	`UserToken`, `AccountID`, `Privilege`, optional `Note`	Gives a privilege to an account that holds none. 409 `Privilege exists` otherwise. (Signup already gives every account `new`, so this follows a `delete`.)
`modify`	Administrator	as `add`	Changes the privilege an account holds. 404 `No privilege` when it holds none.
`delete`	Administrator	`UserToken`, `AccountID`, optional `Note`	Takes the privilege away. The account then reports `none` and cannot log in.
`log`	Administrator	`UserToken`; optional `AccountID`, `Skip`, `Limit`	The audit trail, newest first: `Log[]` with `AccountID`, `Username`, `ChangedBy`, `ChangedByUsername`, `Action`, `OldPrivilege`, `NewPrivilege`, `Note`, `IP`, `Timestamp`.

`Privilege` is one of `administrator`, `contributor`, `user`, `banned`, `new`. `Note` is plain text, max 255 characters, and goes into the audit log.

Two rules protect the site from locking itself out. Nobody can change their own privilege (403) — it takes a second administrator to demote one. And the last administrator can neither be demoted nor deleted (409 `Last administrator`). Every change is written to the audit log, which outlives the accounts it describes. Setting `banned`, or deleting the privilege, revokes every session of the account at once.

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0..Zk3 … ",
  "AccountID": 57,
  "Privilege": "user",
  "Note": "Known scener - accepted"
}
```



```
{
  "message": "Success",
  "subcode": "Privilege modified",
  "submessage": "nightlord: new → user.",
  "AccountID": 57,
  "Username": "nightlord",
  "OldPrivilege": "new",
  "Privilege": "user"
}
```

User Accounts: Setup

> **For whoever runs the server.** Until step 2 is done the four endpoints answer 503 `Not configured` — they fail closed.

1. **Tables.** Run `objects/sql_useraccounts.sql` once against the `RadioAPI` database. It creates `UserAccounts`, `UserProfiles`, `UserProfileGroups`, `UserProfileLinks`, `UserPrivileges`, `UserPrivilegeLog`, `UserSessions`, `UserTokens` and `UserAuthEvents`; the file explains every index. It loads unchanged on MariaDB 11.8 and MySQL 8.4. Table names are case-sensitive on Linux. The legacy `users` table is not used.
2. **Key.** In `config.php`, set `$UserTokenKey` to 32 random bytes, base64-encoded: `php -r "echo base64_encode(random_bytes(32));"`. It is *not* shared with radio.ericade.net. Changing it logs everybody out.
3. **Trusted frontend.** `$UserTrustedFrontendIPRanges` must contain the address radio.ericade.net's PHP reaches the API from — and nothing else. See ClientIP.
4. **Image directory.** `$UserProfileImagePath` (`/var/www/html/radio.ericade.net/images/profiles/`) must exist and be writable by the API's PHP-FPM user. The website's repository ships an `.htaccess` for it that serves nothing but `<32 hex>.webp`. PHP needs GD with WebP support; `exif` is used when present.
5. **Mail.** Sent with PHP's `mail()` from `$FromEmail`. `$UserSiteURL` is where the links point. `$UserMailLogFile` is for development only — it writes mails, tokens included, to a file instead of sending them, and must be empty in production.
6. **First administrator.** Sign up and confirm the address normally, then promote the account by hand — the statement is at the end of the `.sql` file. Every later change is made through privileges and lands in the audit log.
7. **SceneID (optional).** Register the site at id.scene.org with the redirect URI `https://radio.ericade.net/login/sceneid/`, put the client id and secret in `$SceneIDClientID` / `$SceneIDClientSecret`, and set `$SceneIDEnabled = 1` in the website's `config.php`.
8. **Housekeeping.** `PurgeUserAuthData()` runs with the nightly tasks (`sct/tasks.php`): expired sessions and mail tokens a week after they ran out, login attempts after 30 days, and accounts that never confirmed their address a week after their last link expired.

All limits and lifetimes on this page are defaults; they live in `config.php` under "User accounts", each with its own comment.

Content Administration: Overview

Two more look after the catalogue itself: songdata (what the station knows about a tune) and artistdata (the artists' cards). Contributors may read them, administrators change them.

A fourth, message, lets an administrator speak to everybody who is connected right now, over the station data WebSocket.

And action lets an administrator start one of a fixed list of maintenance jobs on the server — the ones it otherwise runs by itself at night.

Last, chat is the little the chat WebSocket cannot do itself: it hands a logged-in visitor a one-time login for the socket, and tells an administrator how the chat server is doing and who wrote what.

Three endpoints let logged-in people change what the station publishes: podcast (episodes), news (its `add` / `modify` / `delete` side — reading news stays where it was, under News & Podcasts) and comment (what visitors write under songs, episodes, artists and news items). They are what the pages under radio.ericade.net/conelrad/ and the comment boxes on the site are built on.

> **Same rules as the user account endpoints.** Everything under How these differ applies here unchanged: `POST` with a JSON body, `Token` on every call, `UserToken` to say who is calling, one envelope (`message` / `subcode` / `submessage` + data), status codes that mean something (400 / 401 / 403 / 404 / 409 / 429), `Cache-Control: no-store`, CORS for radio.ericade.net only, and `ClientIP` / `UserAgent` honoured from the trusted frontend. The action is the rest of the path: `/radio/v2/news/image/add`.

Who may do what

WHAT	PRIVILEGE NEEDED
Podcast episodes: list, candidates, get, add, modify, delete, file, scan, scanstatus, image/add, log	`administrator`
Songs: list, get, log	`contributor` or `administrator`
Songs: modify, delete	`administrator`
Artists: list, prioritized, get, log	`contributor` or `administrator`
Artists: add, modify, delete	`administrator`
News: list, get, preview, add, modify, delete, images, log	`contributor` [1] or `administrator`
Comments: read	None — public, no login
Comments: write; edit one's own for 15 minutes	`user` and up
Comments: edit any, disable, enable (approve), delete, the site-wide list, log	`contributor` or `administrator`
Comments: the two settings (open/closed, moderation)	`contributor` to read, `administrator` to change
Actions: list, run, status, log	`administrator`
Messages: send, preview, log	`administrator` — and, when the server is set up for it, the broadcast automation server (send and preview only)

[1] with the narrower HTML allow-list, see below. `new` and `banned` accounts can do nothing here: an account comments once an administrator has accepted it. The privilege is read from the database on every call, so a demotion or a ban takes effect on the next request.

HTML in news items

`News` is the one field in the API that holds HTML, and the website prints it as it is. It is therefore never stored as sent: it is parsed with an HTML5 parser into a tree, the tree is filtered against an allow-list, and the result is written out again. What is not on the list loses its tags and keeps its text; `script`, `style` (below the top tier), forms, `object`/`embed`, `svg`/`math`, `base`/`meta`/`link` and their kind are removed with everything inside them. The result is then read back by the same parser and refused altogether (400 `Invalid News`) should anything live still be found in it.

TIER	WHO	ELEMENTS	ATTRIBUTES
contributor	`contributor`	`p br h2 h3 h4 ul ol li a b strong i em u s blockquote figure figcaption img hr pre code div span table thead tbody tfoot tr th td caption sub sup small iframe`	`title lang dir`; `href` (http, https, mailto, a path on the site, `#anchor`); `target="_blank"`; `img`: `src alt width height loading`, https on the station's own hosts or a site path; table spans; `ol start`.
administrator	`administrator`	all of the above, plus `style h1 h5 h6 section article header footer aside details summary mark abbr cite q dl dt dd audio video source picture time kbd del ins`	all of the above, plus `class`, `id` (not starting with `sp-`), `style`, `role`, `aria-label`, `aria-hidden`; images from any https host; audio/video from the station's own.

- No tier keeps an event handler (`on…`), a `javascript:`, `vbscript:` or `data:` address, `srcdoc`, or a `data-` attribute.
- `iframe` is kept only for `https://www.youtube.com/embed/<id>` and `https://www.youtube-nocookie.com/embed/<id>`, and gets the API's own `sandbox`, `allow` and `referrerpolicy` — never the ones it arrived with.
- Links get `rel="noopener"` (`noopener noreferrer` with `target="_blank"`).
- CSS (a `style` element or attribute, administrators only) is kept whole or dropped whole. It is dropped when it holds a backslash escape, `@import`, `expression(`, `behavior:`, `-moz-binding`, markup, or a `url()` that is not https or a path on the site.
- Line breaks between elements become spaces; inside `pre` they become `<br>`.

Use news/preview to see exactly what a save would store. An article that holds administrator-tier markup is protected from contributors: see `Administrator markup` under news/modify.

Plain text fields

Titles, blurbs, show notes, playlists and notes are plain text: they are refused (400) when they contain `<` or `>`, a control character or a Unicode direction override, and are otherwise stored as typed. A comment's `Body` is the exception to the first rule — people do write `<3` — and is for that reason *only ever text*: print it escaped.

> The text columns of `titles` and `trackdata` are entity-decoded by the read endpoints (news, songs). These endpoints store their text so that it comes out of that decoding exactly as it was typed; nothing needs to be encoded or decoded by the caller, on either side.

The audit log

Every change made through these endpoints — including every comment posted or edited — writes a row to `ContentAuditLog` in the same transaction as the change. The `log` actions read it, newest first: podcast/log (`podcast`), songdata/log (`song`), artistdata/log (`artist`), news/log (`news`, `image`), comment/log (`comment`, `setting`) and message/log (`message`). They take optional `EntityID`, `Skip` and `Limit` (default 25, max 100) and return `Log[]`:

FIELD	DESCRIPTION	EXAMPLE
`LogID`	Number of the entry.	412
`Entity`, `EntityID`	`podcast` (titles id), `song` (titles id), `artist` (ArtistID), `news` (trackdata trackid), `comment` (CommentID), `image`, `setting`, `message` (always 0).	news, 1015
`Action`	`add`, `modify`, `delete`, `disable`, `enable`, `upload`, `set`; for an episode also `scan` (asked for), `scanned` / `scan-failed` (the worker's result, written as the system) and `image` (cover art uploaded); for a message, its kind: `station-event` or `system-event`.	modify
`AccountID`, `Username`, `Privilege`	Who did it, and what they held at that moment. Copied, so the entry outlives the account. `0` / `(system)` for a message sent by the automation server.	57, nightlord, contributor
`Summary`	One line for a person: the title, the first words of a comment.	Live from Revision
`Details`	Object. For a `modify`: each changed field with `from` and `to`. For a deleted news item: the whole article. For a deleted song or artist: the whole record — it is the only copy from then on. For a comment that was deleted, disabled or edited: its text, its author and the moderator's `Note`. For a message: `Event`, `StationID`, `Severity`, `Source`, `Subject` and `Message` as they were typed, and `Sender` (`administrator` or `automation`) — the only record of it, a message is stored nowhere else.	{"about": {"from": "…", "to": "…"}}
`IP`, `Timestamp`	The caller's address (see ClientIP) and the epoch.	203.0.113.7, 1789654321

Podcast

Podcast episodes. Administrators only.

`POST` `https://api.ericade.net/radio/v2/podcast/<action>`

An episode is a track. A podcast episode is a row of `titles` on the podcast station (StationID 2) with `IsPodcast = 1`. The row itself — title, artist, tags, audio, cue points — is created and kept up to date by the playout's ingest and is read-only here. `add` therefore does not create anything: it *promotes* a track that is already there, and `delete` turns it back into an ordinary track. Nothing is ever inserted or removed. The flag can also be set as a field, `IsPodcast`, of `modify` — see Setting IsPodcast.

ACTION	TAKES	DOES
`list`	optional `Search` (an episode number, or words of the title), `Skip`, `Limit`	`Total` and `Podcasts[]`, highest number first: `TrackID`, `Title`, `IsPodcast`, `EpisodeNumber`, `BroadcastDate`, `isRSS`, `Image`, `PodcastImage`, `PodcastURL`, `UpdatedAt` (the database's own words, in the database's time zone) and `UpdatedEpoch` (the same moment as Unix time — what a page shows in its reader's time zone). **The title decides what is listed, not the `IsPodcast` flag**: every track of the podcast station whose title starts with one to three digits and a dot (`1. Cool stuff`, `153. Chiptunes!!!` — not `123 music` or `My fox 4`). A new recording therefore shows up as soon as the ingest has made its row, with `IsPodcast` 0 and `EpisodeNumber` 0 until `add` (or `modify` with `IsPodcast` 1) has made it an episode.
`candidates`	optional `Search`, `Skip`, `Limit`	`Tracks[]`: the tracks of the podcast station that are not episodes, newest first — `TrackID`, `Title`, `Artist`, `Added`. What `add` can be pointed at.
`get`	`TrackID`	`Podcast`: the fields below plus the read-only `Title`, `Artist`, `Tags`, `StationID`, `IsPodcast`. Works on any track, episode or not. 404 `Not found`.

ACTION	TAKES	DOES
`add`	`TrackID`, `EpisodeNumber`, `BroadcastDate`; any other field	Makes the track an episode. 409 `Already a podcast`; 400 `Wrong station`; 409 `Episode number taken`. `IsPodcast` may be sent as 1 and changes nothing; 0 contradicts the action: 400 `Invalid IsPodcast`. Returns `Podcast`.
`modify`	`TrackID` and the fields to change; optional `IsPodcast` (1 / 0)	A field that is not sent is left alone; one sent empty is cleared — and a value that is neither text nor a number (an array, an object, `true` / `false` for anything but `isRSS`) is refused with 400 rather than read as "empty". 404 `Not a podcast` (the track is not an episode and no `IsPodcast` was sent); 400 `Nothing to change`. With `IsPodcast` it also sets the flag, and then works on any track of the podcast station: Setting IsPodcast. Returns `Podcast`.
`delete`	`TrackID`	Sets `IsPodcast = 0` and `isRSS = 0`: out of the episode lists and the feeds. The track, its audio and the episode data stay — an `add` brings it back as it was. (`modify` with `IsPodcast` 0 does the same and saves the rest of the request as well.) `IsPodcast` 1 contradicts the action: 400 `Invalid IsPodcast`.
`file`	`TrackID`	Is the episode's audio on the server? `File`: `Present` (0/1), `FileName`, `Folder`, `Size`, `Modified` — plus the last `Scan`. See Audio file, scan and cover picture.
`scan`	`TrackID`	Queues a scan of the audio file and answers at once with `Job`. 404 `File absent`; 409 `Already scanning` (with the `Job` that is in the way); 503 `Not available` until the scan's tables exist.
`scanstatus`	`TrackID`	The latest `Job` of the episode (or `null`), `File`, and `Scan` — what to poll while a scan runs.

ACTION	TAKES	DOES
`image/add`	`TrackID`, `FileData` (base64 PNG or JPEG)	Stores the cover picture in every size and format and returns `Image`, `PodcastImage`, `Files[]`, `Width`, `Height`. Does *not* change the episode: send the two addresses with `modify`. 413 `Too large` when the request body exceeds the server's limit.
`log`	optional `EntityID` (a TrackID), `Skip`, `Limit`	The audit log of episodes.

Setting IsPodcast

`modify` takes the flag itself as the field `IsPodcast`: `1` (or `true`, `"1"`) = the track is a podcast episode, `0` (`false`, `"0"`) = it is not. It is what the box "This track is a podcast episode" of the website's editor sends. **Anything else — `"yes"`, `2`, `1.0`, an empty string, an array — is refused with 400 `Invalid IsPodcast` and nothing is written**: 0 takes an episode off the website and out of the feeds, and a value the API does not understand must never be read as 0. Left out (or `null`), `modify` is for episodes only, as it always was. The other fields of the request are validated as ever and saved in the same transaction, whatever the flag becomes; one refused field refuses the whole request.

THE TRACK IS	`ISPODCAST`	WHAT `MODIFY` DOES
an episode	1	Nothing about the flag: an ordinary `modify` (400 `Nothing to change` when no other field was sent).
an episode	0	What `delete` does — `IsPodcast = 0` and `isRSS = 0`, **whatever `isRSS` the request holds**: what is not an episode is in no feed — and the other fields of the request are saved. The episode number, the date and everything else stay on the track. `file`, `scan`, `scanstatus` and `image/add` answer 404 `Not a podcast` from then on.

THE TRACK IS	`ISPODCAST`	WHAT `MODIFY` DOES
not an episode	1	What `add` does, by `add`'s rules: the track has to be on the podcast station (400 `Wrong station`), and an episode has a number and a date — **in the request or on the track already** (a track that was an episode once kept both): 400 `Missing EpisodeNumber` / `Missing BroadcastDate`. The number, sent or stored, must not be another episode's: 409 `Episode number taken` — a stored number may have been given away since. `isRSS` is *not* switched back on by itself: send it.
not an episode	0	Saves the fields on the track and publishes nothing: an episode can be written before it goes out. Podcast station only (400 `Wrong station` — this endpoint is no way to write to the rows of the music catalogue); `isRSS` is stored as 0; an `EpisodeNumber` that belongs to an episode is refused (409). 400 `Nothing to change` when no other field was sent.

The answer is `Podcast modified` in all four cases, with a `submessage` that says what became of the track, and `Podcast` — whose `IsPodcast` is what is stored now; a client that set the flag compares the two. The audit row has the action `modify` and, when the flag changed, `IsPodcast` with its old and new value among the details. A track that stops being an episode is an ordinary track of the podcast station to the rest of the API: songdata reaches it, and the song scan looks at it once (and, finding no MP3 of its own on the website, marks it `skipped`); made an episode again, both let go of it.

Episode fields

FIELD	DESCRIPTION	EXAMPLE
`EpisodeNumber`	0–99999, unique among episodes.	167
`BroadcastDate`	`YYYY-MM-DD`.	2026-09-17
`About`	The blurb. Plain text, max 5000; line breaks become spaces.	Thirty years of Second Reality.
`PlayList`, `ShowNotes`	Plain text, max 60000 each, line breaks kept.	00:00 Intro
`ProductionNotes`	Plain text, max 20000.	Recorded live.

FIELD	DESCRIPTION	EXAMPLE
`Equipment`, `Footer`	Max 5000. The website prints these as HTML, so they pass the allow-list at its narrowest: `a b strong i em u br`. Send lines; they are stored with ` ` between them and come back from `get` as lines.	Shure SM7B
`Image`, `PodcastImage`	A complete `https://` address on one of the station's image hosts (`$ContentImageHosts`). Max 355 / 512.	https://radio.ericade.net/wp-content/uploads/167.webp
`PodcastURL`	The audio file: `https://` on `$ContentMediaHosts`, ending in `.mp3`, `.m4a`, `.ogg` or `.opus`.	https://radio.ericade.net/Flashback/167.mp3
`PodcastFileLength`	Bytes. Left out or 0, it is measured from the file when the file is on the API's server.	81234567
`isRSS`	1 = include the episode in the podcast feeds.	1
`Explicit`	`yes`, `no`, `clean`, `true`, `false`, or empty.	false
`Transcript`	Plain text, max 600000, line breaks kept. A new speaker starts with the name and the time in brackets, which is what the website splits it on. A transcript that is sent back *exactly as* `get` *handed it out* is not judged and not rewritten — old ones hold characters today's rules refuse.	DJ Daemon (12:34) Welcome back.
`Timestamp`	Unix epoch, seconds. Sets the columns `timestamp` and `timestamphr` together — the second is always the first written out (`YYYY-MM-DD HH:MM:SS`, server time) and cannot be sent by itself. Empty = left alone. `get` returns both, as `Timestamp` and `TimestampHR`. Note that the playout's ingest moves both to "now" whenever it plays the episode on the air.	1789598848

Audio file, scan and cover picture

Three things about an episode are files rather than fields. All of them are found from the `EpisodeNumber` *stored on the episode* — never from a number, a name or a path in the request — so they work on saved episodes only (404 `Not a podcast`, 409 `No episode number`).

The audio is `<EpisodeNumber>.mp3`: in the `AmigaFlashback` folder for episodes 1–34, in `Flashback` from 35 on. It is far too large for a form and is put on the server by hand; `file` says whether it has arrived. `get` carries the same `File` object. The folder's name is given, the server's path never is.

A scan decodes the whole file — about a minute per hour of audio — and therefore never runs inside a request. `scan` only queues a `Job`; a worker on the API's server (`sct/podcastscan.php`, started by cron every minute) takes it, reports `Progress` while ffmpeg works, and writes what it found to the episode. Poll `scanstatus` every two or three seconds meanwhile. One scan per episode at a time; a scan of a file changed less than two minutes ago is refused as "still uploading"; a job whose worker died is declared `failed` after ten silent minutes. Scanning again replaces the earlier result.

OBJECT	FIELDS
`Job`	`JobID`, `Status` (`queued` → `running` → `done` | `failed`), `Progress` (0–100, percent of the audio worked through), `Message` (a sentence: what it is doing, or why it failed), `Requested`, `Started`, `Finished` (epochs).
`Scan`	`null` until the first scan. `Status` (`ok` | `failed`), `Timestamp`, `Message`, `Stale` (1 = the file's size or date has changed since: scan again), `FileSize` (bytes — what `PodcastFileLength` should be), `Duration` (s), `Bitrate` (bit/s), `SampleRate` (Hz), `Channels`, `Codec`, and the measurements below; `WaveformURL`, `ID3` (an object of the file's tags), `CoverArt` (1 = the file holds a picture).

MEASUREMENT	WHAT IT IS	READING IT
`LUFS`	Integrated loudness of the whole programme, EBU R128.	Podcast platforms aim for about -16 (stereo).
`LRA`	Loudness range, LU.	How far the quiet and the loud parts lie apart.
`TruePeak`	True peak, dBTP (4× oversampled).	Above -1 may clip once encoded.
`LeftRMS`, `RightRMS`	Level of each channel over the programme, dBFS. `RightRMS` is `null` for mono.	
`ChannelDiff`	Left minus right, dB.	0 = balanced; +3 = the left channel is 3 dB louder.
`StereoSeparation`	Side (L−R) minus mid (L+R), dB.	About -100 = mono in two channels; -20 to -6 = ordinary stereo; 0 and up = unrelated or out-of-phase channels.
`DR`	Dynamic range, dB — the "DR" of the TT Dynamic Range Meter. The recording is cut into blocks of 3 seconds; per channel, DR is the *second* highest block peak minus the RMS of the loudest 20 % of the blocks (RMS raised by 3.01 dB, so that a sine wave measures 0); the file's DR is the mean of its channels. Two decimals, never below 0. `null` when it was not measured: a scan from before 2026-09-19, silence, or fewer than three blocks.	"DR12" is this value rounded. Under 8 = heavily compressed; 8–13 = moderate; 14 and up = very dynamic. The same music turned down measures the same. Chip music made of steady square waves lands low by nature.

Digital silence (-∞) is reported as -99.99. The waveform at `WaveformURL` — `https://radio.ericade.net/images/waveforms/<EpisodeNumber>.webp?v=<time of the scan>` — is a chart, not a bare wave: an opaque, dark WebP of 1800×604 pixels (1800×364 for mono), the left channel above the right, with **the level in dB up the side and the time in hh:mm:ss along the bottom** and grid lines at -3, -6 and -12 dB. The amplitude is linear, as in an audio editor, so the dB lines are not evenly spaced (half way from the middle of a lane to its edge is -6 dB), and every column shows the *loudest* sample that fell into it — the wave reaches the level `TruePeak` says it should. The letters are part of the picture: show it 760 pixels wide or wider and let it scroll sideways on a narrow screen. An episode scanned before the axes were added keeps its old, bare picture until it is scanned again. All of this — without `Stale`, and with its texts HTML-escaped — is also public, as `FileData` in every track answer. **The ID3 tags are text somebody else typed**: cleaned of control characters and cut at 500 characters, but otherwise as they are — escape them wherever they are printed.

The cover picture (`image/add`) is a PNG or JPEG, 150–3000 pixels a side, at most 5 MB, sent as base64 in `FileData`. As with every upload here, what is stored is drawn again from the pixels: not a byte of the upload, its metadata or its name reaches the disk. It goes to `wp-content/uploads/<yyyy>/<mm>/` — the year and month of the episode's `BroadcastDate`, so a new picture replaces the old one where it lies — under the episode's number, as the complete set the site and the feeds link to:

UPLOAD	FULL SIZE	IN THE BOXES 1024×1024, 150×150, 300×300, 768×768
PNG	`167.png`, `167.jpg`, `167.webp`	`167-300x300.png`, `167-300x300.jpg`, `167-300x300.webp`, … (15 files)
JPEG	`167.jpg`, `167.webp`	`167-300x300.jpg`, `167-300x300.webp`, … (10 files; a PNG set it replaces is removed)

Scaling keeps the proportions and only ever shrinks, but every name is written. The answer's `Image` (`…/167.png` or `.jpg`) and `PodcastImage` (`…/167.webp`) are what the episode's fields of the same names should be set to — with `modify`, like any other change.


```
{
  "message": "Success",
  "subcode": "Scan status",
  "submessage": "The latest scan is done.",
  "TrackID": 48211,
  "Job": { "JobID": 31, "Status": "done", "Progress": 100, "Message": "Scanned.", "Requested": 1789745246,
"Started": 1789745290, "Finished": 1789745361 },
  "File": { "Present": 1, "FileName": "167.mp3", "Folder": "Flashback", "Size": 81234567, "Modified":
1789740000 },
  "Scan": {
    "Status": "ok", "Timestamp": 1789745361, "Message": "", "Stale": 0,
    "FileSize": 81234567, "Duration": 5077.29, "Bitrate": 128000, "SampleRate": 44100, "Channels": 2,
    "LUFS": -16.4, "LRA": 6.8, "TruePeak": -1.2, "LeftRMS": -19.8, "RightRMS": -20.1, "ChannelDiff": 0.3,
"StereoSeparation": -12.6, "DR": 9.4,
    "WaveformURL": "https://radio.ericade.net/images/waveforms/167.webp?v=1789745361",
    "ID3": { "title": "167. Thirty years of Second Reality", "artist": "DJ Daemon", "date": "2026" },
"CoverArt": 1, "Codec": "mp3"
  }
}
```

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0..Zk3 … ",
  "TrackID": 48211,
  "EpisodeNumber": 167,
  "BroadcastDate": "2026-09-17",
  "About": "Thirty years of Second Reality.",
  "PodcastURL": "https://radio.ericade.net/Flashback/167.mp3",
  "isRSS": 1
}
```

```
{
  "message": "Success",
  "subcode": "Podcast added",
  "submessage": "\"167. Thirty years of Second Reality\" is now a podcast episode.",
  "Podcast": { "TrackID": 48211, "Title": "167. Thirty years of Second Reality", "IsPodcast": 1,
"EpisodeNumber": 167, "PodcastFileLength": 81234567, " … ": " … " }
}
```

Song Data

What the station knows about a tune. Contributors read, administrators change.

`POST` `https://api.ericade.net/radio/v2/songdata/<action>`

A song is a track the playout has played. It is a row of `titles` that is neither a podcast episode nor a news item — a condition every query here carries, so this endpoint cannot touch an episode however the id is chosen (404). The row is made by the playout's ingest the first time the file is played, matched on its `Guid` and station, and title, artist, album, tags, path, cue points and the play counters are kept up to date from the audio file from then on. So there is **no** `add` — a song exists once the playout has a file for it — and `modify` writes seven columns and no others: what the station knows and the file does not.

ACTION	WHO	TAKES	DOES
`list`	Contributor	optional `Search` (words of the title or the artist, or a TrackID), `StationID`, `Sort` (`newest` (default), `title`, `artist`, `rating`, `plays`), `Skip`, `Limit` (default 25, max 100)	`Total` and `Songs[]`: `TrackID`, `Title`, `Artist`, `StationID`, `CompositeRating`, `Voters`, `TotalPlays`, `Image`, `HasAbout`, the two switches, `UpdatedAt`.

ACTION	WHO	TAKES	DOES
`get`	Contributor	`TrackID`	`Song`: the title first, then the editable fields, then every other column of the track, and `Editable[]` naming the seven. Then what the song scan measured in the song's MP3: `Scan` — the same object as an episode's `Scan`, levels and `DR` included, plain text (not HTML-escaped, unlike the public `FileData`); `Stale` is always 0, the file being on another server; `null` = never scanned — and `ScanQueue`, where the song stands in that job's queue: `Status` (`queued` | `running` | `done` | `failed` | `skipped`), `Attempts`, `Message` (why it failed or was skipped), `Queued`, `Finished`, `NextTry` (epochs); `null` = not in the queue. Both are read-only: nothing in a request starts or changes a scan. 404 for anything that is not a song.
`modify`	Administrator	`TrackID` and the fields to change	A field that is not sent is left alone; one sent empty is cleared. Any other field in the request is ignored; a value that is neither text nor a number (an array, an object, or `true` / `false` for anything but the two switches) is refused with 400 rather than read as "empty". 400 `Nothing to change`. Returns `Song` and `Changed[]` — which also names `MA_NextUpdate` when the ModArchive file name changed.

ACTION	WHO	TAKES	DOES
`delete`	Administrator	`TrackID`	Removes the row, its links to its artists, its uploaded pictures and the waveform the song scan drew for it — the last only when no other row has the same `Guid` (returns `ArtistLinksRemoved`, `PicturesRemoved`, `WaveformRemoved`). **For tracks that have left the library**: a song the playout still has comes back at its next play, as a new row with a new id and none of what was written here. 409 `On the air` when it is playing right now. Its comments are left alone.
`log`	Contributor	optional `EntityID` (a TrackID), `Skip`, `Limit`	The audit log of songs. A deleted song's whole record is kept there.
`image/add`	Administrator	`TrackID`, `FileData` (base64; PNG or JPEG, 150–3000 pixels a side, max 5 MB)	Stores a picture for the song and returns `Image`, `WebP`, `Files[]`, `Width`, `Height`. **Changes no row**: send `Image` to `modify` to start using it. 404 for anything that is not a song; 413 `Too large`.

Editable fields

FIELD	DESCRIPTION	EXAMPLE
`CompositeRating`	Stars, 0–5 with at most one decimal (`4,5` is read as `4.5`).	4.5
`Voters`	Whole number, 0 or more. **Both are worked out again from the listeners' votes every time somebody rates the song**: what is set here lasts until the next vote.	17
`About`	Plain text, max 5000; line breaks become spaces. Refused when it holds `<` or `>`: it ends up in the song page's JSON-LD.	Written for The Party 1992.
`Image`	A complete `https://` address on one of the station's image hosts (`$ContentImageHosts`), max 355 — typically the `Image` that `image/add` returned.	https://radio.ericade.net/images/songs/14906.png
`Explicit`	`yes`, `no`, `clean`, `true`, `false`, or empty.	clean
`isBroadcastProhibited`	1 = we may have it, but it cannot generally be broadcast.	0
`isDownloadProhibited`	1 = cannot be downloaded from the website.	1

FIELD	DESCRIPTION	EXAMPLE
`ModArchiveTrueFileName`	The name the module is filed under on ModArchive.org, for the tracks where that is *not* the name the file has here: the enrichment job (`EnrichFromModArchive`) then looks this name up instead of the one in the album text. A bare file name of at most 200 characters — no folder, none of `/ \ : * ? " < > |`; line breaks and runs of blanks become one blank. Empty = no override. **Changing it also empties** `MA_NextUpdate`, which is how that job is told the track is due: the next run — tonight's, or one started through action — looks the new name up, replaces the `MA_…` data, or clears it when ModArchive has no module of that name. Both changes are in the audit log. The name is stored so that every reader gets back exactly what was typed; a name for which that is not possible (`%20`, a typographic apostrophe) is refused.	virgill-bratgrumbeere.mod

> **A field is only judged when it changes.** A form sends every field back on every save, and the catalogue is full of values from before there were rules — a picture on a host that is no longer allowed, a text with markup in it. A field that arrives *exactly as* `get` *handed it out* is dropped before validation and not rewritten; change it, and today's rules apply. The same holds for artistdata.

Every other field is read-only and comes back decoded — the ingest stores text entity- and percent-encoded — as plain text that must be escaped wherever it is printed: a title is whatever somebody typed into an audio file's tags. Returned are all columns of `titles` except those only episodes and news items use (`podcast`, `podcastimage`, `BroadcastDate`, `Equipment`, `PlayList`, `ProductionNotes`, `EpisodeNumber`, `IsPodcast`, `IsNews`, `PodcastURL`, `isRSS`, `PodcastFileLength`, `Footer`, `News`, `shownotes`, `transcript`, the four checksums and the scan columns): `TrackID`, `Title`, `Artist`, `StationID`, `ArtistID`, `LegacyTrackID`, `Guid`, `TrackIdent`, `Album`, `Genre`, `Year`, `Comments`, `Tags`, `Type`, `TrackerType`, `TrackGroups`, `Path`, `Duration`, `CueIn`, `CueOut`, `OutCue`, `Segue`, `Intro`, `Sweeper`, `NoFade`, `Disabled`, `Added`, `ValidFrom`, `Expires`, `TotalPlays`, `LastPlayed`, `LastPlayedHR`, `Timestamp`, `TimestampHR`, `EligibilityTime`, `ArtistEligibilityTime`, `CreationDate`, `CreationDateHR`, `StartedUtc`, `StartedLocal`, `UpdatedAt`, `mp3present`, `flacpresent`, `originalpresent`, the 23 `MA_…` fields of the ModArchive lookup and `ModArchiveTrueFileName`.

Pictures of songs and artists

`songdata/image/add` and `artistdata/image/add` work as the cover picture of a podcast episode does. The picture is a PNG or JPEG, 150–3000 pixels a side, at most 5 MB (`$PodcastImageMaxBytes`), sent as base64 in `FileData`. What is stored is drawn again from the pixels — not a byte of the upload, its metadata or its name reaches the disk — under the **number** of the song or the artist, which the API reads from its own database: `images/songs/<TrackID>.png|jpg` and `images/artists/<ArtistID>.png|jpg` on radio.ericade.net, each with a WebP copy and the boxes 1024, 150, 300 and 768 in every format — 15 files for a PNG, 10 for a JPEG, the same set as in the table under Scan podcast. No request field can influence the path.

The upload answers with the picture's address and changes no row; `modify` with that address as `Image` does. A new upload replaces the old files under the same names (a JPEG takes the PNG set away), and deleting the song or the artist removes its pictures. Every upload is in the audit log with the files it wrote.

The song scan

Every song gets what only podcast episodes had: the measurements of its audio in `FileData`, a waveform chart, and the four checksums in `Integrity`. **There is no action for it.** It is a background job — `ScanSongs()`, run by `sct/songscan.php` from cron every minute — that works through the catalogue on its own: one song at a time from a queue (the table `SongScanQueue`), so that whatever stops it — a reboot, a killed process, a full disk — costs the song it was working on and nothing else, and the next start carries on where the queue says. The first sweep of a few thousand songs takes days; after it, a new song is done within minutes of its MP3 being published.

The file is the MP3 a listener gets, the one behind `musicURL`: what follows the last backslash of the playout's path, with `.flac` or `.wav` turned into `.mp3` (any other name is used whole), under `https://radio.ericade.net/audio/mp3/`. The audio is on another server than the API, so each file is fetched over HTTPS into a temporary file, measured and checksummed from that one copy, and deleted again. The name travels percent-encoded to that one address; a redirect is not followed, an answer that does not call itself audio is refused, and a file the audio server says was changed less than ten minutes ago (`Last-Modified`) is left for later — a checksum is never worked out twice, so not of half an upload. The measurement is the one Scan podcast makes — the same function, the same columns: loudness, true peak, the level of each channel, stereo separation and the dynamic range. The checksums are MD5, SHA-1, SHA-256 and SHA-384 of the file, as for an episode, but in the table only: no checksum files are published for songs. The chart is stored as `images/waveforms/<Guid>.webp` — the track's `Guid` — where an episode's is `<number>.webp`: five digits at most against thirty-two, so one can never be the other.

What it leaves alone. A song that has both a scan and its checksums is never touched again, and a checksum that is already in the table is kept as it stands; to have a song done again, empty its columns (`objects/sql_songscan.sql` says how). One exception, and only on request: songs that were scanned before the scan measured the dynamic range (2026-09-19) have every number but that one. `php sct/songscan.php --measure-dr`, run once, queues them to be scanned again — whole, behind the songs that have no scan at all, their checksums kept. Nothing does that by itself, and a song whose DR cannot be measured is not taken round and round; while a song is in the queue for another reason and lacks its DR, it is measured then. **Podcast episodes, their data and their files are out of its reach**: every statement it runs carries "not an episode, not a news item", however a `TrackID` got into the queue. A track without an MP3 of its own — a station ID, an episode's recording — is skipped. A missing MP3 (404) is tried again after a day, then two, five times at most; an audio server that does not answer at all ends the run and costs no song an attempt. While an administrator's *Scan podcast* is waiting the job steps aside, and it runs at a lowered priority.

`songdata/delete` takes the song's waveform along (`WaveformRemoved`) unless another row — the same file on the other station — has the same `Guid`. How far along it is: `php sct/songscan.php --status` on the server, and one summary a day in the station's Discord channel while there is something to tell. Setting it up: Setup, "The song scan".

Example: modify


```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0 .. Zk3 … ",
  "TrackID": 14906,
  "About": "Written for The Party 1992.",
  "isDownloadProhibited": 1
}
```

```
{
  "message": "Success",
  "subcode": "Song modified",
  "submessage": "The song has been saved.",
  "Song": { "TrackID": 14906, "Title": "Plastic pop", "Artist": "Dr. Future", "CompositeRating": 4.2,
"Voters": 17, "About": "Written for The Party 1992.", "isDownloadProhibited": 1, "…": "…" },
  "Changed": ["About", "isDownloadProhibited"]
}
```

Artist Data

The artists' cards: descriptions and links. Contributors read, administrators change.

`POST` `https://api.ericade.net/radio/v2/artistdata/<action>`

An artist is two rows. `artists` is what the playout's ingest looks a name up in, and `artistdata` is the card the website shows; `ArtistID` — here as in the public artists endpoints — is the number both share. The ingest makes both the first time it meets a name, and owns the name, the play counters and (through the listeners' votes) the rating. This endpoint edits the card.

ACTION	WHO	TAKES	DOES
`list`	Contributor	optional `Search` (words of the name, or an ArtistID), `Sort` (`name` (default), `songs`, `newest`, `rating`, `plays`), `Skip`, `Limit` (default 25, max 100)	`Total` and `Artists[]`: `ArtistID`, `Artist`, `StationID`, `Songs`, `CompositeRating`, `Voters`, `TotalPlays`, `HasShortDescription`, `HasLongDescription`, `UpdatedAt`.
`prioritized`	Contributor	—	Whose biography to write next: at most 20 `Artists[]` (`ArtistID`, `Artist`, `StationID`, `Songs`) that have **no `LongDescription` and more than one song**, the ones with the most songs first.
`get`	Contributor	`ArtistID`	`Artist`: the name, the editable fields (with `Image` once the table has the column), then every other column of the card (`CardID`, `Slug`, `StationID`, `Guid`, `TotalPlays`, `LastPlayed`, `CompositeRating`, `Voters`, `EligibilityTime`, `UpdatedAt`), `Songs` and `Editable[]`.

ACTION	WHO	TAKES	DOES
`add`	Administrator	`Artist` (the name, 1–120 characters of plain text), `StationID`; any field	Makes the `artists` row and the card, so that a biography can be there before the first tune is played. The name is stored *the way the ingest stores it*, so the ingest finds the artist then instead of making a second one — write it exactly as it stands in the audio files' artist field. For an artist the ingest made without a card, makes just the card. 409 `Artist exists` (with its `ArtistID`).
`modify`	Administrator	`ArtistID` and the fields to change	As songdata/modify: unsent = left alone, empty = cleared, unchanged = not judged, anything else in the request ignored. Returns `Artist` and `Changed[]`.
`delete`	Administrator	`ArtistID`	Removes the card, the `artists` row and the artist's uploaded pictures (`PicturesRemoved`) — and is **refused while any track is credited to the artist** (409 `Artist has songs`, with `Links`): it is for misspellings and leftovers, and can never take a tune's credits away.
`log`	Contributor	optional `EntityID` (an ArtistID), `Skip`, `Limit`	The audit log of artists. A deleted artist's card is kept there.
`image/add`	Administrator	`ArtistID`, `FileData` (base64; PNG or JPEG, 150–3000 pixels a side, max 5 MB)	Stores a picture for the artist and returns `Image`, `WebP`, `Files[]`, `Width`, `Height`. **Changes no row**: send `Image` to `modify` to start using it. 404 for an artist without a card; 503 `Not configured` until `artistdata` has its `image` column; 413 `Too large`.

Editable fields

FIELD	DESCRIPTION	EXAMPLE
`ShortDescription`	Plain text, max 1000.	Finnish composer, member of Future Crew.
`LongDescription`	The biography. Plain text, max 20000, line breaks kept. Both descriptions are refused when they hold `<` or `>` — written out, or as an entity such as `<` that would become one after the decoding some of the public endpoints apply. They are **stored as typed and are text**: escape them wherever they are printed.	Started tracking on the Amiga in 1989…
`Image`	A complete `https://` address on one of the station's image hosts (`$ContentImageHosts`), max 355 — typically the `Image` that `image/add` returned. It is what the public artists endpoints hand out as `Image`. Column `image`, which one `ALTER TABLE` adds: **until it has been run the field does not exist** — it is not returned, not listed in `Editable[]`, and a value that is sent is ignored.	https://radio.ericade.net/images/artists/4821.png
`Demozoo`, `CSDB`, `Pouet`, `ModArchive`, `Bandcamp`, `SoundCloud`, `YouTube`, `Wikipedia`	A complete `http://` or `https://` address, max 255, **on that service's own host** or a subdomain of it: demozoo.org, csdb.dk, pouet.net, modarchive.org, bandcamp.com, soundcloud.com, youtube.com / youtu.be, wikipedia.org. The website prints each link under that service's name.	https://demozoo.org/sceners/42/
`OtherUrl`	The same, on any host — a home page, a Bandcamp page on a domain of the artist's own.	https://example.org/
`isBroadcastProhibited`, `isDownloadProhibited`	0 / 1, as for a song.	0


```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0 .. Zk3 … "
}
```



```
{
  "message": "Success",
  "subcode": "Prioritized artists",
  "submessage": "2 artists.",
  "Artists": [
    { "ArtistID": 4821, "Artist": "Dr. Future", "StationID": 1, "Songs": 14 },
    { "ArtistID": 377, "Artist": "Cube", "StationID": 1, "Songs": 9 }
  ]
}
```

News Administration

Writing news. Contributors and administrators. A news item is a row of `trackdata` with `IsNews = 1` and a `trackid` below 2000; *reading* news is public and stays under News & Podcasts and News by ID, which are unchanged.

`POST` `https://api.ericade.net/radio/v2/news/<action>`

ACTION	TAKES	DOES
`list`	optional `Skip`, `Limit`	`Total` and `News[]`, newest `BroadcastDate` first: `TrackID`, `Title`, `About`, `Image`, `BroadcastDate`, `Created`, `UpdatedAt`, `UpdatedEpoch` (the same moment as Unix time) — without the article.
`get`	`TrackID`	`News`: the same plus `News` (the article's HTML) and `AdminMarkup` (1 = it holds markup only an administrator can write).
`preview`	`News`	Stores nothing. Returns `Html` — exactly what `add` / `modify` would store for this caller — and `Tier`.
`add`	`Title`, `News`; optional `About`, `BroadcastDate` (default: today), `Image`	Publishes at once, under the next free id. 409 `No free id` when the ids below 2000 are used up. Returns `News`.
`modify`	`TrackID` and the fields to change	A field not sent is left alone; `About` and `Image` sent empty are cleared; a value that is neither text nor a number is refused with 400 rather than read as "empty". 403 `Administrator markup` when a contributor sends `News` for an article whose `AdminMarkup` is 1 — saving it would strip the layout; its other fields can still be changed. Returns `News`.
`delete`	`TrackID`	Removes the item and the comments under it (`CommentsDeleted`). The whole article goes into the audit log.
`image/add`	`FileData` (base64; JPEG, PNG, GIF or WebP, max 4 MB)	Stores the picture and returns `FileName`, `URL`, `Width`, `Height`. Use `URL` as `Image`, or in an `<img>` in the article.

ACTION	TAKES	DOES
`image/delete`	`FileName`	Removes an uploaded picture. 409 `Image in use` while a news item refers to it.
`log`	optional `EntityID`, `Skip`, `Limit`	The audit log of news items and images.

FIELD	DESCRIPTION	EXAMPLE
`Title`	Plain text, 1–200 characters, no `<` or `>`: it ends up in the page's `<title>` and structured data as well as its heading.	Live from Revision
`About`	The blurb the lists and link previews show. Plain text, max 1000; line breaks become spaces. Column `about`.	We broadcast from the party hall all weekend.
`News`	The article: HTML, max 250 kB, filtered through the allow-list for the caller's privilege. 400 `Invalid News` when nothing is left of it. Column `News`.	Friday …
`BroadcastDate`	`YYYY-MM-DD`.	2027-04-02
`Image`	A complete `https://` address on one of `$ContentImageHosts` — typically the `URL` from `image/add`.	https://radio.ericade.net/images/news/3f9c…e1.webp

An uploaded picture is never stored as uploaded. The same pipeline as profile images: size judged on the base64, dimensions read from the header before anything is decoded, then decoded, turned upright, scaled to at most 1600 pixels wide and re-encoded as WebP under a random 32-hex name in `$ContentImagePath`. No request field can influence the path; `image/delete` accepts nothing but such a name.


```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0 .. Zk3 … ",
  "Title": "Live from Revision",
  "About": "We broadcast from the party hall all weekend.",
  "BroadcastDate": "2027-04-02",
  "News": "<h2>Friday</h2><p onclick=\"x()\">Doors open at <strong>noon</strong>.</p><script>x()</script>"
}
```



```
{
  "message": "Success",
  "subcode": "News added",
  "submessage": "The news item has been published.",
  "News": {
    "TrackID": 1015,
    "Title": "Live from Revision",
    "About": "We broadcast from the party hall all weekend.",
    "Image": "",
    "BroadcastDate": "2027-04-02",
    "Created": 1789654321,
    "UpdatedAt": "2026-09-17 21:14:02",
    "UpdatedEpoch": 1789679642,
    "News": "<h2>Friday</h2><p>Doors open at <strong>noon</strong>.</p>",
```

```
    "AdminMarkup": 0
  }
}
```

Comment

What visitors write under a song, a podcast episode, an artist or a news item.

`POST` `https://api.ericade.net/radio/v2/comment/<action>`

A comment hangs on a target: `TargetType` is `song`, `podcast`, `artist` or `news`, and `TargetID` the TrackID, ArtistID or news TrackID. `song` and `podcast` are one kind of target (both are tracks, and a track that becomes an episode keeps its comments). A comment is `visible`, `pending` (waiting for a moderator — only while moderation is on) or `disabled` (hidden by a moderator, kept).

ACTION	WHO	TAKES	DOES
`list`	Anybody	`TargetType`, `TargetID`; optional `UserToken`, `Skip`, `Limit` (default 50, max 100), `Moderation`	The visible comments under the target, newest first: `Total`, `Settings`, `CanPost`, `CanModerate`, `Comments[]`. With a `UserToken`, the caller's own `pending` comments ride on top of the first page; a moderator who sends `Moderation: 1` gets every status.
`list`	Contributor	`UserToken`, *no target*; optional `Status`, `Skip`, `Limit`	The moderation queue across the site: `Counts` per status, and each comment with `TargetType`, `TargetID` and `TargetTitle`.
`add`	User	`UserToken`, `TargetType`, `TargetID`, `Body`	Posts a comment — shown at once, or `pending` while moderation is on (never for a moderator's own). 403 `Comments closed`; 404 when the target does not exist; 409 `Duplicate`; 429 `Too many comments` (5 in ten minutes; moderators exempt). Returns `Comment`.

ACTION	WHO	TAKES	DOES
`modify`	Author / Contributor	`UserToken`, `CommentID`, `Body`	The author, for 15 minutes after posting (403 `Too late`) and not once it is disabled; a moderator, always. Somebody else's comment answers 404. While moderation is on, an author's edit sends the comment back to `pending`. Marks it `Edited`.
`disable`	Contributor	`UserToken`, `CommentID`, optional `Note`	Hides it, keeps it. The gentle alternative to deleting.
`enable`	Contributor	as `disable`	Shows a disabled comment again, or approves a pending one.
`delete`	Contributor	as `disable`	Removes it for good. Its text is kept in the audit log. Authors cannot delete their own comments.
`settings`	Contributor (read), Administrator (change)	`UserToken`; optional `CommentsEnabled`, `CommentModeration` (0 / 1)	Returns `Settings`. `CommentsEnabled: 0` closes commenting site-wide (reading still works); `CommentModeration: 1` makes new comments wait for approval. Defaults: open, unmoderated.
`log`	Contributor	optional `EntityID` (a CommentID), `Skip`, `Limit`	The audit log of comments and settings.

The Comment object

FIELD	DESCRIPTION	EXAMPLE
`CommentID`	Number of the comment.	812
`Body`	The text, with the profanity filter applied. **Plain text: escape it wherever it is printed.** 2–2000 characters; line breaks are kept.	What a f****** tune!
`Status`	`visible`, `pending` or `disabled`.	visible
`Created`, `Edited`	Epochs. `Edited` is 0 when the text was never changed.	1789654321, 0
`Author`	`DisplayName` (scene handle, else name, else user name), `ProfileImageURL`, `Privilege`. Moderators also get `AccountID` and `Username`.	{"DisplayName": "Nightlord"}

FIELD	DESCRIPTION	EXAMPLE
`Own`, `CanEdit`	1 when the comment is the caller's / when the caller could `modify` it right now.	1, 1
`BodyRaw`	The text as typed. Only for the author and for moderators — an edit form needs it.	What a fucking tune!
`IP`, `ModeratedBy`, `Moderated`, `ModerationNote`	Moderators only.	203.0.113.7

Profanity filter

A basic one: the words on `$CommentProfanityList` (English and Swedish), as whole words in any case and with the usual endings, keep their first letter and get asterisks for the rest. *Scunthorpe* is left alone; creative spelling gets through, which is what `disable` and moderation are for. The filter is applied when a comment is *read* and the text is stored as typed, so the list can change afterwards and moderators can see what was really written.

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0 .. Zk3 … ",
  "TargetType": "song",
  "TargetID": 48211,
  "Body": "Still gives me goosebumps. <3"
}
```

```
{
  "message": "Success",
  "subcode": "Comment added",
  "submessage": "Your comment has been posted.",
  "Comment": {
    "CommentID": 812,
    "Body": "Still gives me goosebumps. <3",
    "Status": "visible",
    "Created": 1789654321,
    "Edited": 0,
    "Author": { "DisplayName": "Nightlord", "ProfileImageURL":
"https://radio.ericade.net/images/profiles/9a1c … e0.webp", "Privilege": "user" },
    "Own": 1,
    "CanEdit": 1,
    "BodyRaw": "Still gives me goosebumps. <3"
  }
}
```

Message

Sends words, rather than track data, over the station data WebSocket: a `station-event` for listeners, which radio.ericade.net shows on every open page, or a `system-event` for administrators and their tools, which the website ignores. Administrators only. This is what radio.ericade.net/conelrad/message/ posts to.

`POST` `https://api.ericade.net/radio/v2/message/<action>`

> **Both events are public.** The socket has no login, so whoever connects to it reads every `system-event` as well: "the website does not show it" is not "secret". Never put an address, somebody's name or e-mail, a path, a token or the text of an exception in a message. Nothing about the sender is broadcast either: a message carries its severity, its subject, its text and the name it is signed with — the account, its privilege and its address stay in the audit log.

ACTION	WHO	TAKES	DOES
`send`	Administrator	`UserToken`, `Event`, and `Subject` or `Message` (or both); optional `Severity`, `StationID`, `SendAs`	Broadcasts the message, at once, to every client that is connected; it is not stored and not replayed to clients that connect later. `POST /radio/v2/message` without an action is `send`. Returns `Event`, `StationID`, `StationName` and `Payload` — the payload exactly as it went out. 409 `Duplicate` for the same message twice within 15 seconds; 429 `Too many messages` (6 a minute per sender, with `Retry-After`); 503 `Not available` when the WebSocket relay is switched off or cannot be reached — nothing was sent, and nothing is logged as sent. **It cannot be taken back.**
`preview`	Administrator	as `send`	Validates exactly as `send` does and returns the same answer with `Preview: 1` — what *would* go out, after cleaning. Sends nothing, logs nothing, and is not counted against the limit.

ACTION	WHO	TAKES	DOES
`log`	Administrator	`UserToken`; optional `Skip`, `Limit`	The audit log of sent messages, newest first: who, when, from where, and the text as it was typed.

Fields

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Event`	Yes	`station-event` (for listeners) or `system-event` (for administrators' tools).	Exactly one of the two, in lower case. 400 `Invalid Event`.
`Severity`	No	`information`, `low`, `medium`, `high` or `critical`.	Default `information`. Case does not matter. 400 `Invalid Severity`.
`StationID`	No	`0` = the whole network: every client receives it, whatever it subscribed to. Otherwise a station, and the clients following it.	Default `0`. 400 `Invalid StationID`.
`Subject`	No[1]	One line of plain text, at most 120 characters.	Default `Station message` / `System message`. 400 `Invalid Subject`.
`Message`	No[1]	Plain text, at most 4096 characters. Line breaks are kept.	Default empty. 400 `Invalid Message`.
`SendAs`	No	`station`: the message is from the station — `Source` is the server's `$StationMessageSource`, "LeisaWolfe", the broadcast automation server. `me`: it is signed with the caller's public `DisplayName`, the name already shown next to their comments.	Default `station`. `Source` is never free text: nobody speaks in somebody else's name. 400 `Invalid SendAs`.

[1] one of the two must hold something: 400 `Empty message`. Characters are counted as characters, not bytes — an emoji is one.

What happens to the text

`Subject` and `Message` are plain text; `<` and `>` are allowed ("uptime > 99 %") because nothing here is ever markup. Control characters and Unicode direction overrides are refused (400). On its way out every message then passes one function, `BuildStationMessagePayload()` — the same one the server's own code uses through `PublishStationMessage()`:

- Invalid UTF-8 becomes U+FFFD; every kind of line break becomes a line feed; three or more in a row become two.
- **IPv4 and IPv6 addresses are replaced by `[address removed]`.** These events are read by anybody, and the classic way an address ends up in one is a variable somebody put into a status line. A four-part version number goes the same way; write "version 8.5".
- Last, `Subject`, `Message` and `Source` are **HTML-escaped**: `& < > " '` travel as `& < > &quot; &#039;`. A client that carelessly uses `innerHTML` therefore still shows text. See Displaying the text for what a careful client does.

Use `preview` to see the result before anything is sent.

The broadcast automation server

The playout (“LeisaWolfe”) can send messages without a user login — “Silence detector: audio is back.” — when the server has been set up for it. **It is off by default.** Such a call carries `Token`, `AutomationKey` and *no* `UserToken`, and is accepted only when the key matches `$StationMessageAutomationKey` (32 characters or more) *and* the call really comes from an address in `$StationMessageAutomationIPRanges`. The address is the connection’s own; `ClientIP` is not considered. Every other case answers the same 403. Its messages are always from the station (`SendAs: me` answers 400), it shares the limits above, it may `send` and `preview` but not read the `log`, and its messages are logged with no account and `Sender: automation`.

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0..Zk3 … ",
  "Event": "station-event",
  "Severity": "medium",
  "Subject": "Maintenance tonight",
  "Message": "The stream moves to a new server at 22:00 CET.\nExpect a short break <3"
}
```

```
{
  "message": "Success",
  "subcode": "Message sent",
  "submessage": "The message has been sent to everybody who is connected right now.",
  "Event": "station-event",
  "StationID": 0,
  "StationName": "",
  "Preview": 0,
  "Payload": {
    "Severity": "medium",
    "Source": "LeisaWolfe",
    "Subject": "Maintenance tonight",
    "Message": "The stream moves to a new server at 22:00 CET.\nExpect a short break &lt;3"
  }
}
```

> The `Payload` in the answer is the one on the wire: note the `<` where `<` was typed.

Action

Starts a maintenance job on the API's server: the script that empties the web server's caches, and the jobs the server otherwise runs by itself at night — for the day somebody does not want to wait for the night. Administrators only. This is what radio.ericade.net/conelrad/action/ is built on.

`POST` `https://api.ericade.net/radio/v2/action/<action>`

Nothing is run in the request, and nothing in a request says what to run. `run` takes the *name* of an action, compares it — exactly, as a string — with the names of a list that is part of the API's code, and puts a row in a queue. The name is never part of a path, a function name or a command, and no other field of the request reaches the job. A worker on the server (root's crontab, once a minute) takes the queue, looks the name up in the same list *again* — so that not even a row planted in the table can start anything else — and runs the job in a process of its own. **Anything that is not on the list is refused with 400 and written to the action log as a warning**; so is a logged-in caller who is not an administrator (403). A request without a login is refused without a row: anybody can send those, and the log is not theirs to fill.

ACTION	TAKES	DOES
`list`	—	`Actions[]` — every action on the list with its state, the job of it that is waiting or running (`Current`) and its latest finished one (`Last`), both as a job or `null` — and the state of the whole: `Active` (jobs queued or running), `BusyLimit`, `LockedUntil` / `LockedSeconds`, the two lengths `CooldownSeconds` and `LockSeconds`, `WorkerSeen` (when the worker last started) and `WorkerLate` (1 = not for three minutes: jobs can be queued, but nothing will run them), `Now`.
`run`	`Action` — a name from the list below	Queues the job and returns it as `Job`, with `Active`, `LockedUntil` and `LockedSeconds`. It starts within a minute; jobs run **one after another**, oldest first. 400 `Invalid Action` (and a warning in the log) for anything that is not on the list; 409 `Already queued` / `Already running` (with that `Job`) while a job of the same action is; 429 `Cooling down` or 429 `Locked` — see limits — with `WaitSeconds`, `WaitReason` and a `Retry-After` header.

ACTION	TAKES	DOES
`status`	`JobID`	One job, as `Job`. 404 when there is none.
`log`	optional `EntityID` (a JobID), `Skip`, `Limit`	The audit log of actions: `queued` (who, from where), `done` / `failed` (written by the worker, as the system), `locked`, and every `warning`.

The actions

ACTION	WHAT IT DOES	WHO
`CleanCache`	Clears the web server's caches and resets the PHP-FPM OPcache. (A script on the server, run without a shell — and only while it and its directory belong to root and nobody else can write to them. When it is not run, the job's `Message` says *which* of those it was — not found, not executable, not root's, writable by others — and what mends it, without naming the path; the error log has the path, the owner and the mode.)	Administrator
`CreatePodcastIntegrity`	Checks all podcast episodes and generates hash signatures for them — the `Integrity` object and the checksum files next to each MP3.	Administrator
`EnrichFromModArchive`	Checks whether any files need new data from ModArchive.	Administrator
`ValidateTrackPresence`	Goes through all songs and podcast episodes to see which audio files exist in the store.	Administrator
`ConvertToWebp`	Checks that all images have thumbnails and WebP versions.	Administrator
`PopulateCueFiles`	Creates new .cue files for the podcast episodes still missing them.	Administrator
`PopulatePodcastTunes`	Makes sure the songs played in podcast episodes are sent to their reference table.	Administrator
`PurgeOldEpisodes`	Removes songs that are no longer played on the station (not played for 21 days). **Deletes**: `Destructive` is 1 in `list`, and the website asks "are you sure?" first.	Administrator

The names are exact and case-sensitive. `list` is the authority: a client should offer what it returns and nothing else.

Limits

- **One job of an action at a time.** While one is queued or running, `run` for the same action answers 409.
- **A cooldown of 5 minutes** (`$ActionCooldownSeconds`) from the *end* of a job until the same action can be started again — 429 `Cooling down`. Other actions are not affected.
- **More than 7 jobs at the same time** (`$ActionBusyLimit`; queued or running) **locks everything for 10 minutes** (`$ActionLockSeconds`) — 429 `Locked` for every action. The job that made it eight is still taken, and its answer says that it locked the door behind it; jobs that are already queued still run.

- A job that runs longer than 4 hours (`$ActionTimeoutSeconds`; `CleanCache`: 10 minutes) is stopped and ends as `failed`. So does one whose worker died: after ten minutes without a sign of life.

State of an action

FIELD	DESCRIPTION	EXAMPLE
`Action`, `Description`, `Privilege`, `Destructive`	The name, a sentence for a person, who may start it, and 1 when it deletes something.	CleanCache
`State`	`idle` (can be started), `queued`, `running`, `cooldown`, or `locked`.	cooldown
`CanRun`	1 when `run` would be accepted right now.	0
`WaitSeconds`, `WaitReason`	How long until it can be started, and why: `cooldown` or `locked`. 0 and empty otherwise.	212
`QueuePosition`	1 = next, for a queued job. 0 otherwise.	2
`Current`, `Last`	The job that is queued or running, and the latest finished one. `null` when there is none.	Object{}

A job

FIELD	DESCRIPTION	EXAMPLE
`JobID`, `Action`	The job's number and the action it is a run of.	412
`Status`	`queued`, `running`, `done` or `failed` (it crashed, ran out of time, never reported back, or the script was missing or not safe to run).	done
`Outcome`	What the job said about *its own* run: `ok`, `warnings` or `errors`; empty when it said nothing. A job can be `done` with `errors`: it ran to the end and found that three episodes have no audio file.	ok
`Message`	One sentence for the administrator. Never a path or the text of an exception — those are in the server's error log.	Finished in 2 min 14 s.
`Report`	The job's own summary — what it sends to Discord, or what the script printed — as plain text of at most 4000 characters, tags taken out. **It is a program's text, not markup: escape it wherever it is printed.** It may name a log file on the server, which is why only administrators get it.	12 converted, 3 already up to date…
`RequestedBy`	User name of the administrator who started it.	erik
`Requested`, `Started`, `Finished`, `Seconds`	Unix times (0 = not yet), and how long it ran.	1789745361

```
{
  "Token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 … ",
  "UserToken": "eyJhbGciOiJkaXIiLCJlbmMiOiJBMjU2R0NNIiwidHlwIjoiSldUIn0 .. Zk3 … ",
  "Action": "ConvertToWebp"
}
```

```
{
  "message": "Success",
  "subcode": "Job queued",
  "submessage": "ConvertToWebp has been queued. It starts within a minute.",
  "Job": { "JobID": 412, "Action": "ConvertToWebp", "Status": "queued", "Outcome": "", "Message": "Waiting
for the worker to start",
           "Report": "", "RequestedBy": "erik", "Requested": 1789745361, "Started": 0, "Finished": 0,
"Seconds": 0 },
  "Active": 1,
  "LockedUntil": 0,
  "LockedSeconds": 0
}
```

Chat

The chat on radio.ericade.net — one conversation with the channel #general of the station's Discord — is a WebSocket, `wss://api.ericade.net/ws/chat`, and has a reference page of its own: logging in, the messages, the limits, the Discord bridge, how to run it. These three actions are the little a socket cannot do itself.

`POST` `https://api.ericade.net/radio/v2/chat/<action>`

> **A ticket says who, and nothing else.** The visitor's login is an HttpOnly cookie on radio.ericade.net that the socket can never see. So the website's PHP asks `ticket` with the visitor's `UserToken`, the page presents the ticket as its first message on the socket, and the chat server looks up the account the ticket was made for. What that account may do — write, read along, nothing — is read from the database by the chat server at that moment, again for every message, and every half minute after: not from the ticket, and not from anything the socket is told. `AccountID` in a request to these actions changes nothing.

ACTION	TAKES	DOES
`ticket`	—	A one-time login for the socket, for any logged-in account (`new` and up). See below.
`status`	—	Administrators: is the chat server up, how full is it, is Discord connected. See below.
`log`	optional `DiscordMessageID`, `WriterID`, `Skip`, `Limit`	Administrators: what was written from the website, and by which account. See below.

ticket

For any logged-in account: `new` accounts get one too, because they may read along — whether an account may write is the chat server's to say, and `CanSend` only tells the page what to draw. `banned` accounts and accounts without a privilege cannot log in at all, so they have no `UserToken` to ask with (401).

FIELD	MEANING
`Ticket`	64 hexadecimal characters. Good for **one** login, within `ExpiresIn` seconds. Only its SHA-256 is stored (`ChatTickets`).
`ExpiresIn`, `Expires`	60 seconds, and the same as Unix time.
`WebSocketURL`	Where the socket is (`$ChatWebSocketURL`).
`Name`	The name the account's messages will carry: `Daemon [ericade web] (Administrator)` — the public display name, never the login name.
`Privilege`, `CanSend`	The account's privilege, and `1` when it may write (`user` and up).
`MaxLength`	The longest message the chat server accepts, in characters.

503 while `$ChatEnabled` is 0 or `objects/sql_chat.sql` has not been run. 429 with `Retry-After: 60` for the eleventh ticket of an account within a minute.

```
{
  "message": "Success",
  "subcode": "Chat ticket",
  "submessage": "Present the ticket on the WebSocket within 60 seconds. It is good for one login.",
  "Ticket": "5f0c … 64 hexadecimal characters … a91e",
  "ExpiresIn": 60,
  "Expires": 1789772187,
  "WebSocketURL": "wss://api.ericade.net/ws/chat",
  "Name": "Daemon [ericade web] (Administrator)",
  "Privilege": "administrator",
  "CanSend": 1,
  "MaxLength": 500
}
```


status

What the chat server says about itself (its `GET /status`, asked over the loopback with a one-second timeout), and how the chat is set up. No secret is in the answer: whether Discord is configured, never with what.

FIELD	MEANING
`Enabled`, `SchemaReady`	`$ChatEnabled`, and whether the two tables exist.
`Running`	`1` when the chat server answered.
`Connections`, `Available`, `MaxConnections`	Sockets in use, room left, and `$ChatMaxClients` (200).
`LoggedIn`	Connections that have presented a valid ticket.
`Discord`, `WaitingForDiscord`	`connected`, `not reachable`, `not configured` or `unknown` (server down); and the messages not yet delivered to Discord.
`DiscordReads`, `DiscordWrites`	`1` when a bot token and channel / a valid webhook address are configured.
`Throttle`, `LogRetentionDays`, `WebSocketURL`	The settings in force.

log

In Discord every message from the website is posted by one webhook, under the name its writer chose to show — Discord cannot say which *account* it was. This can: `DiscordMessageID` is what “Copy Message ID” gives in Discord. `WriterID` is an `AccountID`: everything that account wrote. (It is not called `AccountID` on purpose — that field says whose account a call *works on*; this is a filter on a list.) Newest first, 50 a page, at most 200.

FIELD OF `LOG[]`	MEANING
`LogID`, `Timestamp`	The row, and when. The message’s ID on the socket was `web–<LogID>`.
`AccountID`, `Username`, `DisplayName`, `Privilege`	Who wrote it, as the account was at that moment.
`IP`	The visitor’s address, as the proxy reported it to the chat server.
`Message`	The text **as typed** — before the word filter, which works on what others get to see. It is text: whoever prints it escapes it.
`DiscordMessageID`	The id Discord gave it. Empty: it never reached Discord (the website’s users still saw it).

Only what was written **from the website** is here; what is written in Discord is Discord's to keep. Rows are deleted after `$ChatLogRetentionDays` (30) by the chat server. 400 for a filter that is not digits.

Content Administration: Setup

> **For whoever runs the server.** The user accounts must be in place first: these endpoints are nothing without a login.

1. **Tables.** Run `objects/sql_content.sql` once against the `RadioAPI` database, after `sql_useraccounts.sql`. It creates `Comments`, `ContentAuditLog` and `SiteSettings` (with the two comment switches in their default position) and explains every index. It loads unchanged on MariaDB 11.8 and MySQL 8.4. Podcasts and news need no new tables.
2. **Index (recommended).** `titles` has no index on `IsPodcast`; the file ends with the one `ALTER TABLE` that gives the episode lists and the "is this episode number taken" check an index to use. Optional — run it at a quiet moment.
3. **PHP.** The HTML allow-list needs PHP 8.4 or later with the DOM extension (`php-xml`) for its HTML5 parser. There is deliberately no fallback to the older parser: without it, saving HTML answers 400 and the reason is in the error log. Uploaded pictures need GD with WebP, as profile images do.
4. **Image directory.** `$ContentImagePath` (`/var/www/html/radio.ericade.net/images/news/`) must exist and be writable by the API's PHP-FPM user. The website's repository ships an `.htaccess` for it that serves nothing but `<32 hex>.webp`.
5. **Hosts.** `$ContentImageHosts`, `$ContentMediaHosts` and `$ContentEmbedHosts` in `config.php` decide which addresses pictures, audio and embeds may point at.
6. **People.** Give the people who write news `contributor` through privileges/modify (or the website's `/useradmin/` page). Accounts must be accepted (`new` → `user`) before they can comment.
7. **Catalogue and scans.** Run `objects/sql_catalog.sql` once, after the two files above: it creates `PodcastScanJobs`, adds the eighteen nullable `Scan…` columns to `titles` (one `ALTER` — a rebuild of the table, so pick a quiet moment), then a nineteenth, `ScanDR` for the dynamic range — section 2b, a single `ALTER TABLE `titles` ADD COLUMN `ScanDR` DECIMAL(5,2) DEFAULT NULL AFTER `ScanStereoSeparation`` that is run on its own where the eighteen are in already (until it has been, both scans go on measuring everything else and `DR` is `null`; afterwards an episode gets its DR from *Scan podcast again*, the songs from `php sct/songscan.php --measure-dr`) — and lists four *recommended* indexes on existing tables, each with the reason and the `SHOW INDEX` to run first. It loads unchanged on MariaDB 11.8 and MySQL 8.4. songdata and artistdata need no new tables; the artist's picture needs one new column, `artistdata.image` — section 3 of the file, a single `ALTER TABLE `artistdata` ADD COLUMN `image` VARCHAR(355) DEFAULT NULL` that can be run on its own if the rest went in earlier. The code asks the database what is there: until the file has been run the podcast endpoint works without its scan (`ScanAvailable: 0`, `scan` answers 503), every track's `FileData` says `none`, and artists have no `Image`.
8. **The scan worker.** `apt install ffmpeg`, then one line in root's crontab — the same way the WebSocket relay's watchdog is started: `* * * * * /usr/bin/php /var/www/html/api.ericade.net/sct/podcastscan.php >> /var/log/podcastscan.txt 2>&1`. A run that finds nothing queued costs one `SELECT`; only one instance works at a time. `$PodcastWaveformPath` (`/var/www/html/radio.ericade.net/images/waveforms/`) is where it writes; the website's repository ships an `.htaccess` for it that serves nothing but `<number>.webp` (an episode) and `<Guid>.webp` (a song).
9. **Cover pictures.** `$PodcastUploadsPath` (`…/wp-content/uploads/`) and the month folders under it must be writable by the API's PHP-FPM user; missing month folders are created. The 5 MB limit (`$PodcastImageMaxBytes`) is 6.7 MB as base64: going higher means raising `post_max_size` for the API, and `upload_max_filesize` / `post_max_size` for the website, which must allow 5 MB to begin with (PHP's default `upload_max_filesize` is 2M).
10. **Pictures of songs and artists.** `$SongImagePath` and `$ArtistImagePath` (`/var/www/html/radio.ericade.net/images/songs/` and `…/images/artists/`) must exist and be writable by the API's PHP-FPM user; the website's repository ships an `.htaccess` for each that serves nothing but `<number>.png|jpg|webp` and its scaled versions. Both lie inside the tree the nightly WebP job scans, which

is intended: an upload writes every file that job would, so it finds nothing to do. The same 5 MB limit applies.

11. **The letters on the waveform.** `$PodcastWaveformFont` is a TrueType file, by default `/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf` (`fonts-dejavu-core`, which `ffmpeg` brings along). Without it the chart is drawn with GD's built-in letters — coarser, nothing fails. `ffmpeg` 4.4 or later is needed for the wave to show peaks; an older one still scans, and says so in the result.
12. **The song scan.** Run `objects/sql_songscan.sql` once, after `sql_catalog.sql`: it creates one table, `SongScanQueue`, with one index, and changes nothing that exists — no column and no index on `titles`, which the playout writes to. It loads unchanged on MariaDB 11.8 and MySQL 8.4; until it has been run the worker does nothing and says so once an hour. Then one more line in root's crontab, next to the scan worker's: `* * * * * /usr/bin/php /var/www/html/api.ericade.net/sct/songscan.php >> /var/log/songscan.txt 2>&1`. Nothing has to be installed: it uses the `ffmpeg` of the podcast scan and PHP's curl, writes to the same waveform folder, and needs to reach `$SongAudioURL` (`https://radio.ericade.net/audio/mp3/` — `https` only) and room for one MP3 at a time in the system's temporary folder. A run works for `$SongScanRunSeconds` (3000) and ends; cron starts the next within the minute, and only one instance ever works. `$SongScanEnabled = 0` switches it off; `$SongScanPauseSeconds`, `$SongScanSettleSeconds`, `$SongScanMaxAttempts`, `$SongScanDownloadTimeoutSeconds` and `$SongScanMaxFileBytes` are explained in `config.php`. `php sct/songscan.php --status` says how far along it is and what failed last; `--retry-failed` queues the failures once more; `--measure-dr` queues the songs that were scanned before the dynamic range was measured. See The song scan.
13. **Actions.** Run `objects/sql_actions.sql` once: it creates `ActionJobs` (the queue, and the record of every run) with its two indexes, and adds two rows of state to `SiteSettings`. It loads unchanged on MariaDB 11.8 and MySQL 8.4; until it has been run, action answers 503 and nothing else notices. Then one line in root's crontab, next to the scan worker's: `* * * * * /usr/bin/php /var/www/html/api.ericade.net/sct/actionworker.php >> /var/log/actionworker.txt 2>&1`. It has to be root: the jobs write their logs under `/root`, and `sct/tasks.php` runs the same functions as root every night. `$ActionCleanCacheScript` (`/usr/sbin/cacheclean.sh`) is only run while the file *and* its directory belong to root and nobody else may write to them — otherwise whoever can write there decides what root runs; the job then ends as `failed` and the error log says why. The limits are `$ActionCooldownSeconds`, `$ActionBusyLimit`, `$ActionLockSeconds`, `$ActionTimeoutSeconds` and `$ActionMaxParallel` (1 = one job after another). **What can be run is not a setting**: it is the list in `ActionCatalogue()`.
14. **Chat.** Run `objects/sql_chat.sql` once: it creates `ChatTickets` and `ChatLog`, each with three indexes, and changes nothing that exists. It loads unchanged on MariaDB 11.8 and MySQL 8.4. Then the Apache lines, the two Discord values and the watchdog's crontab line, step by step, in the chat's own reference. No package has to be installed. Until the file has been run `chat/ticket` answers 503 and the chat server turns logins away.
15. **Messages.** message needs no table of its own (it writes to `ContentAuditLog`) but does need the WebSocket relay: with `$WebSocketEnabled = 0` it answers 503. `$StationMessageSource` is the name unsigned messages carry, `$StationMessageMaxPerMinute` the limit per sender. To let the broadcast automation server send, set *both* `$StationMessageAutomationKey` (make one with `php -r "echo bin2hex(random_bytes(32));"`) and `$StationMessageAutomationIPRanges` (its address, never this server's own or the web server's); left empty, only administrators can.

All limits on these pages are defaults; they live in `config.php` under "Content administration", each with its own comment.

Ingest

> **Internal endpoint.** Restricted to the internal ingest IP ranges and authenticated with a pre-shared secret rather than a JWT. Documented for completeness; not callable by third parties. Requests from other addresses are rejected with HTTP 400 and "Your IP is not in the allowed range".

Set Next Up

Replaces the station's upcoming queue with the supplied track. Called by the playout automation when it schedules the next item: it clears every existing queue row for the station, resolves the track from its GUID and inserts the replacement. Read the queue with Next Up instead.

`POST` `https://api.ericade.net/radio/v2/nextup`

ATTRIBUTE	MANDATORY?	DESCRIPTION	NOTES
`Password`	Yes	Pre-shared ingest secret. This is not a JWT.	
`StationID`	Yes	The station whose queue is being replaced.	1 = ericade.radio 2 = Best of ericade.radio
`Guid`	Yes	PlayIT Live GUID of the track. Used to resolve the TrackID, artwork, rating and added-date from the library. Max 120 characters.	6894ac27-abc9-4c4a-aad2-15879dda4dff
`Artist`	No	Artist name. Max 255 characters. Replaced by the library value when the track is a listener request.	MA2E
`Title`	No	Track title. Max 255 characters.	Thrilled 4 noise

At least one of `Artist`, `Title`, `Guid` or `StationID` must be supplied. The response is the standard `message` / `subcode` / `submessage` envelope; on success `submessage` is `Track added to next up.`

> A `next-up` event fires on the Station Data WebSocket once the queue is updated.

Other ingest endpoints

The remaining playout-chain endpoints have not been ported to v2 and still live under `/radio/`. See the v1 documentation for `updatetrack`, `updatestreamevent`, `getnextrequest`, `updateplayedrequest`, `checkifexists`, `getnowplaying` and `getlistening`.