Remote Audio Data (RAD)

V2 Draft Spec

This document should be considered an “Internet-Draft”, as defined by the IETF, therefore it may be updated, replaced or obsoleted by other documents at any time and should not be cited other than as "work in progress.”

Background

Documentation for V1 of the Remote Audio Data (RAD) spec can be found on the NPR Dev Center. V2 of the RAD spec is under development in collaboration with the working groups that formed out of the RAD Winter Summit in February 2018. General questions and feedback can be submitted via this form. This document outlines a revised V2 draft specification for a custom ID3 tag, referred to as a Remote Audio Data (RAD) tag. The RAD tag can be encoded with listening event metadata to allow clients to report valuable listening data to podcast content producers and sponsors.

About this Document

This is a working document, intended to solicit edits, suggestions and contributions from participants in the RAD working groups. Participants are encouraged to add comments, and edit in “suggesting” mode. Comments, questions and edits will be reviewed and resolved during working group meetings.

Table of Contents

Goals for RAD V2

1. Gain consensus among publishers, podcast app developers and technology companies on the minimum requirements necessary to implement RAD
2. Solve for business concerns, including governance, GDPR and privacy
3. Define the technical requirements needed to support the creation of client-side RAD libraries
4. Determine a foundational standard for measuring podcast listening events

Fundamental Requirements for RAD Implementation

1. Audio file published with custom ID3 tag that includes encoded metadata that describes RAD-formatted listening events and a tracking URL.
   Note: The encoded ID3 tags will not affect audio playback for any non-RAD clients.
2. RAD-enabled client to deliver audio file to end user.
3. Client-constructed URLs to send data on listening events to the provided tracking URL.
4. Server configured to received constructed URLs from RAD-enabled clients, and make this data available for analysis.

Please see the Appendix and the NPR Dev Center for more detail on the RAD life cycle, as piloted for V1.

Security Requirements

All RAD data requests and responses should use the JSON data interchange format and are required to be sent over HTTPS. The HTTPS requirement also includes any publisher-provided tracking URLs and RSS feeds.

RAD metadata is encoded into ID3 tags, which may be readable by anyone who chooses to inspect the tag within the audio file. Publishers that desire an additional layer of security may choose to encrypt the JSON payload before encoding their RAD metadata into the ID3 tag, and then decrypt the returned data when it is received by their analytics server.

Anonymized identifiers should be used to identify a unique listening session, in place of any user identifiers like an advertising ID or UUID. Clients are advised to include an encrypted string, based on the listening event timestamp, that has been hashed to prevent collision with other reported events. The salted hash should be appended to the constructed URL to serve as a session identifier that the client sends back to the analytics server; this string should suffice to connect multiple events to a discrete listening session without compromising the privacy or security of end users. Please see example below under Client Considerations and Format.

Open questions

- Should a public whitelist for compliant clients and publishers be maintained? If so, what level of governance can this group contribute to maintain this type of whitelist?
- What, if any, approval will be needed for an organization to be added to the whitelist?
- What defines a “good” versus “bad” actor? How will bad actors be identified? Will a blacklist approach be used for those who violate sponsorship playback requirements?

Event Requirements and Format

A series of listening events are saved as objects within the `"events":[ ]` array of the encoded RAD metadata:

```
{
   "events":
   [
     {
       "label":"podcastDownload",
       "eventTime":"00:00:00.000",
       "adId":0,
       "creativeId":0,
       "adPosition":0,
       "eventNum":0
     }
}
```

RAD event labels can be customized to fit the needs of a publisher; examples might include a percentage of completed listening, segment breaks or advertising spots. The string values provided for the `"label"` field can be customized to fit the needs of the publisher, provided that they fit the following conventions:

- Values assigned to event labels should be string types so that non-integer identifiers can be used, such as compound keys or GUIDs.
- No more than XX events should be created per one audio file.
- Event names should be published in camelCase format, and not exceed XX character limit.
- Event parameters should be limited to XX number.

Open questions

- Should any specific events be required? (e.g. `podcastStart, podcastEnd`)
- Should we publish standardized event names for frequently used content and ad event types?
- How should clients handle files that have more than the maximum number of events?
- Should we include tests in a (potential) open source library to test the RAD payload to determine if it is in or out of bounds for the defined data size?
- How should clients handle playback actions such as skip, fast-forward, rewind or repeat listening? Do we need to implement callbacks for user actions like start/stop in addition to publisher-defined event labels?

Standardized Error Codes

Errors can occur in the transmission of RAD data from the client or to the tracking URL. Appropriate HTTP status codes should be used in response to client requests, and clients should respond with the agreed upon course of action:

- 200 (okay request)
- 206 (partial request)
- 404 (bad data type given)
- 500 (server error)

Open questions

- What level of standardization for error codes and error messages should be required?
- What is the desired course of action for a client to take after receiving a 500 series error?
- Should we support a 300 rpoedirect code as well?

Audio Formats

The RAD spec currently supports implementation for MP3 and MP4 audio formats.

Open questions

- Should V2 address support for HLS and HTML5?

Client Considerations and Format

RAD-enabled clients will be required to:

- Parse the payload of the encoded RAD metadata in the audio file's ID3 tag
- Implement a client-side observer to monitor audio playback for listening events
- Construct a URL to send the event data to a remote server for tracking purposes (this may also include implementing a temporary storage solution to allow the client to batch events into fewer callbacks to the tracking URL)

An open source client-side library may be provided at some point to encapsulate the batching and transmission of the RAD listening events to the analytics server.

The constructed URL will include the following parameters appended to the base tracking URL:

- `applicationId`
- `sessionStart (in the form of anonymized hash)`
- `podcastId`

- `episodeId`
- `label`
- `eventTime`
- `timestamp`
- `eventNum`
- Additional parameter key/value pairs, as provided in the event object.

Session ID

As noted above under Security Requirements, anonymized identifiers should be used to identify a unique listening session, in place of any user identifiers, such as an advertising ID or UUID.

Clients are advised to include an encrypted string, based on the listening event timestamp, that has been hashed to prevent collision with other reported events. Please see below for one possible implementation method written in Kotlin:

```
fun sessionHash(): Long {
    var hash = 1125899906842597L

    val string = Date().toString()
    val characters = string.toCharArray()

    val len = string.length
    for (i in 0 until len) {
        hash = 31 * hash + characters[i].toLong()
    }

    return hash
}
```

Calling `sessionHash()on` Wed Apr 25 14:54:42 EDT would return the hashed value of `2018-4960313116615288515`

The salted hash should be appended to the constructed URL to serve as a session identifier that the client sends back to the analytics server (see example below).

Example constructed URL sent as a GET request to the provided tracking URL:

```
"https://tracking.publisher.org/remote_audio_data?applicationId=org.client.podcastdatametrics&sessionStart="sessionHash"&timestamp="timestamp"&podcastId=510298&episodeId=497679856&label=podcastDownload&eventTime=00:00:00.000&adId=0&creativeId=0&adPosition=0&eventNum=0"
```

Offline listening

Open questions

- Is some level of offline support a requirement for a RAD-enabled client?
- What is the expectation for retaining data for events that take place during offline listening?
- What is the recommended method for temporary storage and transmission of offline listening events?

Android

Android clients must first locate the RAD metadata atom within the audio file before the data can be parsed: `[moov] [trak] [udta] [meta] [ilst] [----] [data] "RAD"`

Notes on NPR Android implementation

During the pilot of RAD V1, NPR One used the ExoPlayer 2's MetadataUtil class to parse the metadata from this tag, but the specifics of this implementation will vary based on platform. For NPR One's implementation, our developer needed to remove a check in the MetadataUtil class to avoid returning null early in the function, and allow the client to extract the payload. For interested devs, the line in question can be found here: https://github.com/google/ExoPlayer/blob/d979469659861f7fe1d39d153b90bdff1ab479cc/library/core/src/main/java/com/google/android/exoplayer2/extractor/mp4/MetadataUtil.java#L296 (Line 296-299 of MetadataUtil.java)

NPR One implemented RAD for MP4 audio files, as that is the preferred / default format used in NPR One; the MetadataUtil class is in the mp4 package, so it's unlikely that the MP3 implementation uses it. There have been a few minor releases / changes to ExoPlayer v2.x since the RAD pilot in spring 2017, but MetadataUtil looks relatively unchanged. Should this spec be ratified, we can / should either open pull requests with the ExoPlayer project to remove this check from the MetadataUtil or have a specific namespace added to the ID3 v2 spec, ideally one that falls outside of these omitted namespaces.

iOS

The iOS media player supports the parsing of custom fields in the ID3 tags by default, so no extra effort is needed on the part of the client to locate and parse the metadata encoded into a tag within the file.

Any future client-side library for iOS clients will likely build on the AVFoundation framework; any clients implementing custom players will need to customize their implementation of this library accordingly.

Open questions

- What type of client libraries should be provided? Should there be an SDK? How would this type of open source tool be managed and maintained?
- What method(s) should be in place to verify correct client implementation?

Data Tracking and Analytics

RAD is not intended to replace download statistics as a point of measurement for the on-demand audio industry, rather the reported listening events will complement this metric.

Publishers must provide a tracking URL where clients can report events, but they may choose their own method for storing and analyzing this client-reported RAD data. A tracking URL may point to an in-house analytics solution or to a solution provided by another partner. By grouping listening events by unique identifiers, the data consumer can determine their own analytics methods and protocols.

Publishers may download the reported data to an analytics solution, which may involve a text parser that classifies listening events by download in a queryable form. Ideally, the solution would allow for querying listening events by podcast download and for querying specific listening events across all downloads. A complementary database with podcast information would allow for more complicated queries.

Open questions

- What type of guidelines should be provided for sending and receiving batch events?
- Should we provide a spec for logging RAD data?
- Do we need to specify expectations for the payload size the server will/can accept?
- Should we provide guidance and/or a specification for setting up the required analytics server?
- What new consumption metrics does RAD need to define?

Tracking URL Requirements and Format

A valid tracking URL must be included in the encoded ID3 tag so that a RAD-enabled client can callback to the server with listening event data. Tracking URLs will not be visible to the standard audio consumer, however publishers should be aware that they can be seen to anyone who inspects a ID3 tag.

Publishers may wish to include more than one tracking URL in the encoded ID3 tag so that a client can share listening event data directly with their ad servers or creative partners.

While publishers may configure listening events to track advertising playback, it is not currently in scope to include third-party tracking pixels from advertisers. Tracking URLs should be limited to publishers or vendor(s) designated by the publisher.

Recommendations for server setup

NPR's RAD V1 pilot used existing web server resources, configured with an Apache virtual host entry similar to the following:

```
<VirtualHost *>
  ServerName tracking.example.org

  CustomLog /var/logs/apache2/tracking.example.org-access.log
combined
  ErrorLog /var/logs/apache2/tracking.example.org-error.log

  DefaultType image/gif

  RewriteEngine on
  RewriteRule .* - [R=204,NS,L]
</VirtualHost>
```

Specific server configurations and CPU requirements will depend upon expected traffic and any preexisting system setup.

Open questions

- What type of support will be required for Podtrac or similar redirects?
- What is the maximum number of tracking URLs that should be supported? At what point would requiring clients to send RAD data to multiple URLs become a barrier to implementation?

Privacy Considerations

V2 of the RAD spec does not direct clients or publishers to include personal information in the transmitted data. In place of a device identifier such as an Apple UUID, the Apple IDFA, or the Android Advertising ID, anonymized identifiers should be used to identify a unique listening session. Clients are advised to include an encrypted string, based on the listening event timestamp, that has been hashed to prevent collision with other reported events. Clients and publishers should adhere to applicable law, such as the EU's General Data Protection Regulation's data processing, privacy by design, and privacy by default requirements (to the extent a client or publisher determines it is subject to these requirements).

For Future Discussion

Review by governing bodies

In order for RAD to become a mature and fully-adoptable standard, it is necessary that there be a governing body that manages validation, specification updates and vendor accreditation.

Open Questions:

- Should this be limited to the IAB or should other governing bodies be considered (e.g. MRC, W3C)?

Validation

Open questions

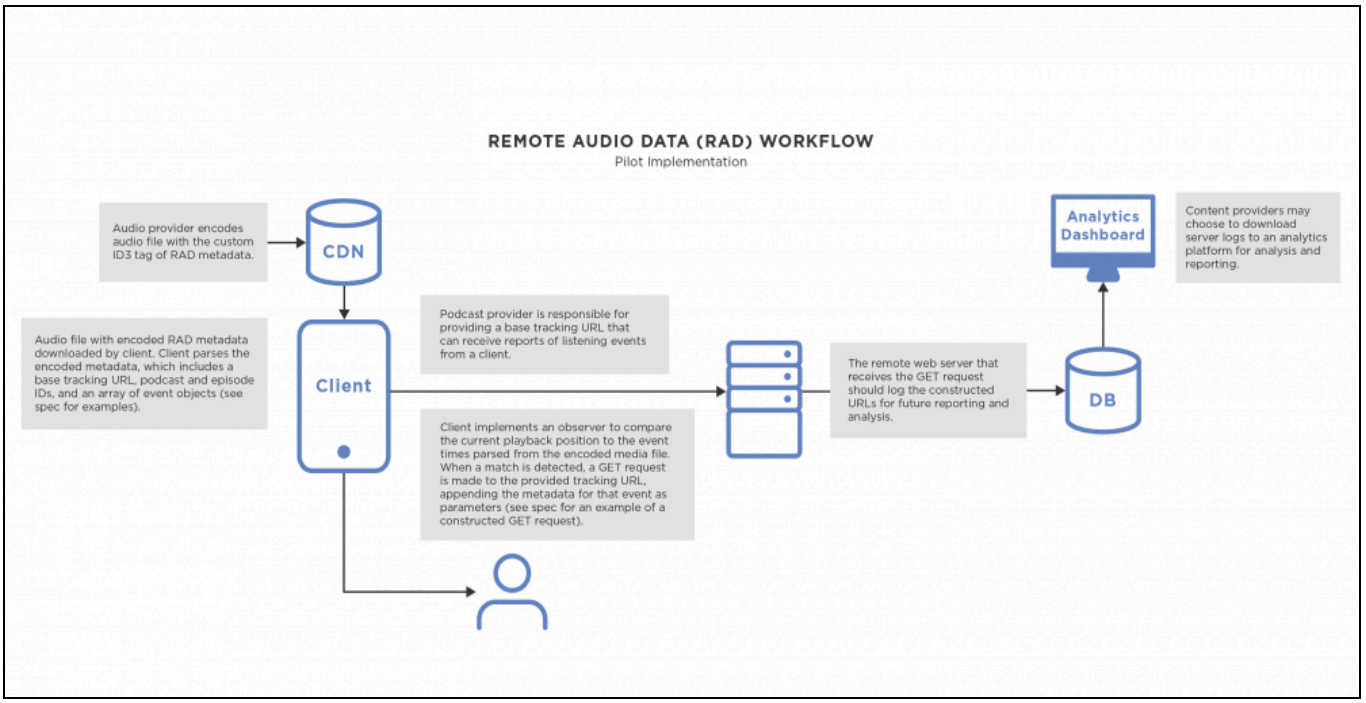
- What type of validation should be put in place to ensure publishers and clients adhere to RAD specifications?
- Should this process be covered under an existing governing body, or one that is newly created to be specific to RAD?

Appendix

RAD V1 Documentation

https://dev.npr.org/rad-spec

RAD V1 Pilot Workflow


REMOTE AUDIO DATA (RAD) WORKFLOW
Pilot Implementation
Audio provider encodes audio file with the custom ID3 tag of RAD metadata.
CDN
Analytics Dashboard
Content providers may choose to download server logs to an analytics platform for analysis and reporting.
Audio file with encoded RAD metadata downloaded by client. Client parses the encoded metadata, which includes a base tracking URL, podcast and episode IDs, and an array of event objects (see spec for examples).
Client
Podcast provider is responsible for providing a base tracking URL that can receive reports of listening events from a client.
The remote web server that receives the GET request should log the constructed URLs for future reporting and analysis.
DB
Client implements an observer to compare the current playback position to the event times parsed from the encoded media file. When a match is detected, a GET request is made to the provided tracking URL, appending the metadata for that event as parameters (see spec for an example of a constructed GET request).