Radioplayer Ingest HTTP API Client User Guide

Audience

The audience for this document is intended to be technical and addresses the concerns of using HTTP as a transport for XML-formatted metadata.

Background

The Radioplayer system currently allows stations to send their XML metadata as files using FTP transport. While this approach provides an easy mechanism for both automated and manual submission of data it provides poor support for automated checking of the state of processing of data.

To address this an HTTP API has been developed that will allow stations to submit their metadata and check the processing state through combination of POST and GET requests.

Introduction

The HTTP API allows authenticated clients to use a POST request to submit their meta-data and supports clients that wish to track the status of their requests using subsequent GET requests.

Each POST request is authenticated and the data checked for compliance to the XML schema. Failures at this point in processing are reported immediately to the client.

Also, the API employs rate-limiting and in the event that you have too many requests pending processing the API may respond with an error code indicating that you should retry your request at a later time.

Once a request has been authenticated, validated and passed through rate-limiting the data is enqueued for asynchronous processing. The results of this asynchronous processing are reported through the subsequent GET request mentioned above.

API Usage

In order to use the API you will need an HTTP client that is capable of HTTPS communication, pre-emptive authentication, setting request headers, reading response status codes and reading the response body.

https

The API requires the use of simple HTTPS in all communication.

Pre-emptive Authentication

The API requires credentials to be submitted with each request.

Metadata Update Request

The API allows clients to submit XML data in the same format that is supported by the FTP-based ingest mechanism. The ingest server requires that clients use an HTTP POST method when performing updates.

For each of the following data types a distinct URL is used to perform the update request.

Service Information (SI)

https://ingest.radioplayer.ca/ingestor/metadata/v1/si/

Programme Information (PI)

https://ingest.radioplayer.ca/ingestor/metadata/v1/pi/

Programme Event (PE)

https://ingest.radioplayer.ca/ingestor/metadata/v1/pe/

Off-Schedule On Demand (OD)

https://ingest.radioplayer.ca/ingestor/metadata/v1/od/

Now Playing (NP)

The API also allows clients to submit a concise form of now-playing data that does not use an XML document to communicate the content. Simple request parameters are used instead of an XML document.

https://ingest.radioplayer.ca/ingestor/metadata/v1/np/

The request parameters are as follows

- rpId - the station's Radioplayer Id
- startTime - the start time of the now playing data in ISO8601 format, UTC timezone
- duration - the duration of the song in seconds
- title - song title, max length 128 characters
- artist - artist name, max length 128 characters
- description (optional) - max length 180 characters
- imageUrl(optional) - Url to an image which will be shown in the search result, must be 86 x 48 pixels

A template example of how you could post this data using the commonly found 'curl' command line utility follows:

```
curl -u yourusername:yourpassword -v --data
"rpId=YOUR_RPID&startTime=2013-10-01T08:27:00&duration=600&title=SONG
TITLE&artist=ARTIST" -X POST
"https://ingest.radioplayer.ca/ingestor/metadata/v1/np/"
```

Read response

After each request the client must read the status code and the response from the ingest server. The status codes that the server may respond with are as follows

202 Accepted

This indicates that the request has been accepted for asynchronous processing. The response body will contain the URL that the client may use to determine the status of the request.

503 Retry-After

This indicates that the request exceeds the rate of allowed update requests and that you must retry the request. The time at which you should retry the request is expressed in **seconds** as an offset from the time you received this response and is provided to your client in a response header called ‘Retry-After’.

Request Processing Status

Once your request has been accepted you may make further requests for the processing status. The URL on which you can find the processing status is provided in the response header called ‘Location’.

Status Tracking

The status of your ‘Accepted’ request can be tracked using a GET request to the URL provided in the response header ‘Location’. The response to this status tracking request is a JSON string showing the latest status of processing.

Example:

```
{
  "id":"261702",
  "timestamp":"30-Mar-2012 17:17:04 BST",
  "protocol":"Http",
  "type":"PI",
  "scope":" RpIds:308  startTime:2012-01-20T00:00:00Z,  stopTime:2012-01-21T00:00:00Z",
  "downstreamStatuses":[
    {
      "downstream":"solr",
      "status":"accepted"
    }
  ]
}
```

Lifecycle of a request

A request received from client to server will go through the following stages or statuses

- Accepted - Request is authenticated and validated successfully and queued for asynchronous processing by downstream systems.
- Processed - Request is successfully processed by downstream systems.

- Failed - Request failed during processing, due to an error.
- Timedout - Request is timedout, as there is no response from downstream after a certain period of time.

Status of meta-data POST requests will only be available for **x** minutes after the processing is completed (success/failure). If the request is not getting processed after **y** minutes, then it is considered as a timedout request and will be purged after **x** minutes from then.