Airtime: Scheduled Audio Playout for Broadcast Radio

Paul Baranowski
CTO, Sourcefabric
720 Bathurst St. #203
Toronto, ON M5S 2R4
paul.baranowski@sourcefabric.org

Abstract

We present Airtime[1], a new open source web application targeted at broadcast radio for automated audio playout. Airtime's workflow is adapted to a multi-user environment found at radio stations with program managers and DJs. Airtime is written in PHP and Python and uses Liquidsoap to drive the audio playout.

Keywords

Radio, broadcast, scheduled playout, web apps.

1 Introduction

Airtime is an open source web application targeted at radio stations who need to automate part or all of their audio playout. Its first release happened on Valentine's Day February 2011.

Airtime is funded by Sourcefabric, a non-profit organization dedicated to making the best open source tools for independent journalism around the world. One of it's primary missions is to support independent media in developing democracies. Sourcefabric is currently funded by grants and has guaranteed funding for at least two more years. Within that time we expect to become self-sustaining.

In this paper we present a common workflow found at radio stations, then present how Airtime's workflow matches that model. We then cover a number of non-workflow based features as well as the technology used to build both the web interface and backend player. We finish up with a preview of future development.

2 Radio Station Workflow

We have designed the interface workflow in a way that many multi-person radio stations work. The two roles present in radio stations related to Airtime are program managers and DJs. Program managers are responsible for organizing the schedule for the DJs and making sure that the schedule is fully booked. They usually plan out the schedule weeks or months in advance. DJs are responsible for preparing and presenting the audio during their assigned time slots("time slots" are also known as "shows"). If the show is live, quite often DJs will bring their own equipment for playout such as turn tables, CDs, or iPods. If the show is automated, the DJ has the responsibility to fill in their show with audio.

3 Airtime Overview

Before we present the Airtime workflow, we present a few of the key concepts in the application: shows, playlists, and roles.

3.1 Shows

A "show" in Airtime corresponds to a block of time allocated to a DJ. It is also a container for audio clips. Shows can be assigned to one or more users, in which case only those users are able to modify the audio within that show. It is possible to create repeating shows on a daily, weekly, bi-weekly, or monthly basis.

3.2 Playlists

Airtime also has playlists, which can be inserted into a show. Playlists can be created before the shows have been scheduled and can be reused. Playlists and shows are completely separated – if a user schedules a playlist inside a show and then deletes the playlist, the schedule still has it's own copy of the song list and playout will not be affected.

3.3 Roles

Airtime has three roles: admin, host, and guest. The "admin" role corresponds to the program manager job; this role has the ability to add, change, or delete shows. They also have the rights of a DJ.

The "host" role is equivalent to a DJ. They have the ability to create playlists and schedule them within the shows they have been assigned.

The "guest" role is a read-only role that allows someone to log in and see what is going on without being able to change anything.

4 Airtime Workflow

The expected workflow for Airtime works as follows: the program manger logs in under the admin role and creates the shows in the calendar for all the DJs. Repeating shows can be scheduled on a daily, weekly, bi-weekly, or monthly basis. The interface in the calendar is very similar to Google Calendar, where the user has the ability to move shows around by drag and drop as well as resize shows with the mouse to change their length.

The DJs log in at their leisure, upload their audio, use the audio library to create playlists, and add their playlists to a show. Any uploaded audio files are automatically scanned for metadata and additional metadata is retrieved from online databases. Replay gain is calculated on the audio files to normalize the output volume.

A status area at the top of the screen displays what song and show is currently playing along with timing and progress information. A more detailed list of the upcoming audio tracks can be viewed on the "Now Playing" screen, which also allows you to see the full list of planned audio for any given day. Any breaks of silence are displayed in red.

Shows that have already played cannot be removed, as this information is typically needed for various regulation purposes.

The backend audio player looks to see what show is scheduled for a specified time and starts playing it. It is completely disconnected from the web interface in that it fetches all the information it needs via HTTP requests and downloads a copy of the music it needs to play.

5 Non-workflow Features

The non-workflow features available in Airtime are internationalization and live show recording.

5.1 Internationalization

The Airtime interface can be internationalized into any language.

5.2 Show Recording and Archiving

Airtime ships with a separate application that hooks into Airtime's schedule which will record the audio during a live show if the user requests it. The audio is saved to a file, and inserted back into the audio database with metadata attached. These audio files can then be replayed again in future shows.

6 Technology

Airtime is written in PHP using the Zend Framework and Propel as the ORM layer. The web interface makes heavy use of jQuery and various jQuery plugins. The playout engine is Liquidsoap controlled by Python scripts. By default we output to both the sound card via

ALSA and to an Icecast stream. We currently only support the Linux operating system at the moment, which is mainly due to the fact that Liquidsoap is primarily supported on *UNIX platforms.

6.1 Design of the Playout System

The scripts used to drive Liquidsoap are collectively called "pypo" for Python PlayOut. These scripts were developed in conjunction with Open Broadcast in Switzerland. There are three separate processes which drive the playout:

1. Liquidsoap
2. Pypo-fetch
3. Pypo-push

Liquidsoap is an open source programmable audio stream engine for radio stations. It expects to always be playing something. We have written custom Liquidsoap scripts to drive the playout based on what Airtime users have scheduled. The Liquidsoap developers have been kind enough to add functionality for our playout model.

Pypo-fetch is responsible for fetching the playout schedule and downloading the music tracks before playout starts. There are configuration values for how far in advance to start downloading the audio as well as how long to keep the audio after the playout has occurred.

Pypo-push is responsible for controlling Liquidsoap and switching the playlist at the right time. It connects to Liquidsoap via a local telnet connection and switches between playlists using the queuing technology found in Liquidsoap.

Each of these programs is installed as a daemon via daemontools under a separate Linux user named "pypo".

7 Future Development

The first release of Airtime has been made for one narrowly defined use case. In the coming year we are planning to develop the additional functionality shown below.

7.1 Very Near Term (3 months)

7.1.1 Scheduling Webstreams

The ability to automatically connect to webstream at a certain time and rebroadcast it.

7.1.2 Jingle Support

Users have requested a quick and easy way to add jingles to a playlist.

7.1.3 AutoDJ (Smart/Random Playlists)

Automatically generate playlists based on certain criteria.

7.1.4 RDS Support

RDS is the technology that displays the name of the song on your radio.

7.2 Mid-term (3-6 months)

7.2.1 Advertising Support

We plan to make Airtime understand the difference between ads and songs. The advertising manager will be able to put ads in the schedule with time boundaries within which those ads must be played. Ads will have different rights than audio and cannot be removed by someone without "advertising manager" rights.

7.2.2 RESTful API

Allow 3rd party applications to get the data out of the database via a REST interface. This would allow others to create other views of the data, such as a Web widget which would display the currently playing audio and display the upcoming schedule.

7.2.3 Playlist Import/Export

This is the ability to export a playlist to a file and import it back in.

7.2.4 Airtime/Newscoop Integration

Newscoop is Sourcefabric's enterprise newsroom software. Integrating with this would allow a station to run it's web site and control it's playout with an integrated set of tools.

7.2.5 SaaS Hosting

We plan on offering a hosted version of Airtime.

7.3 Longer Term (6 months – 1 year)

7.3.1 Live Shows

We are planning to support live shows by allowing 3rd party playout software to access the audio files through a FUSE filesystem. We are also planning on implementing a "live mode" in the browser to allow a DJ to play songs on-demand.

7.3.2 Graphical Crossfading Interface

Display the waveform for an audio file in the browser and allow the user to drag and drop the crossfade points with their mouse and preview it.

7.3.3 Smartphone/Tablet Interface

Allow users to create playlists and schedule them on their favorite smartphone or tablet.

7.3.4 Networked Stations

Allow stations to share content with each other.

8 Conclusion

Airtime is under active development by three developers, a graphic designer, a QA engineer, and a manager. We are engaged with radio stations around the world to listen to feedback and make the most useful project possible. Since it is open source, outside developer participation is welcome in the project. You can try out Airtime right now be going to the demo site[2].

References

[1] Link to Airtime homepage: sourcefabric.org/en/products/airtime_overview/

[2] Airtime demo site: airtime-demo.sourcefabric.org