D3.4: HRADIO Lab Radio playout system specification

Editors: Alexander Erk, Andreas Bosl (IRT)

This Deliverable describes the HRADIO lab playout system, which enables HRADIO project partners to easily set up and operate a DAB+/IP-Streamed radio service for demonstration and development purposes.
### Basic Information

<table>
<thead>
<tr>
<th>Work package</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due date</td>
<td>31/01/2019</td>
</tr>
<tr>
<td>Submission date</td>
<td>25/02/2019</td>
</tr>
<tr>
<td>Deliverable lead</td>
<td>IRT</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Authors</td>
<td>Alexander Erk (IRT), Andreas Bosl (IRT)</td>
</tr>
<tr>
<td>Reviewers</td>
<td>Klaas Baert (VRT), Remo Vogel (RBB)</td>
</tr>
</tbody>
</table>

### Document Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description of change</th>
<th>List of contributor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>20/01/2019</td>
<td>Initial version</td>
<td>Andreas Bosl (IRT)</td>
</tr>
<tr>
<td>V0.8</td>
<td>11/02/2019</td>
<td>Version for review</td>
<td>Andreas Bosl (IRT), Alexander Erk (IRT)</td>
</tr>
<tr>
<td>V0.9</td>
<td>20/02/2019</td>
<td>Final draft after internal review</td>
<td>Klaas Baert (VRT), Remo Vogel (RBB)</td>
</tr>
<tr>
<td>V1.0</td>
<td>25/02/2019</td>
<td>Final version after coordinator review</td>
<td>Simon Dolaere (imec)</td>
</tr>
</tbody>
</table>
Disclaimer

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 731677. This document reflects only the authors’ views and the Commission is not responsible for any use that may be made of the information it contains.

<table>
<thead>
<tr>
<th>Project co-funded by the European Commission in the H2020 Programme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nature of the deliverable:</strong></td>
</tr>
<tr>
<td><strong>Dissemination Level</strong></td>
</tr>
<tr>
<td><strong>PU</strong></td>
</tr>
<tr>
<td><strong>CL</strong></td>
</tr>
<tr>
<td><strong>CO</strong></td>
</tr>
</tbody>
</table>
The main goal of the HRADIO project is to demonstrate the benefits of hybrid radio services for end users and their potential for broadcasters in strengthen the position of linear radio services, when combining them intelligently with non-linear and interactive content elements. However, in order to develop and demonstrate such a combined service, it is often hard to find publicly available “on Air” radio stations, which deliver the exact set of service and program metadata that is required for the demonstration of the full potential of the HRADIO system. For this reason, the tasks in work package 3 include the development of a HRADIO lab playout system, which is capable to fill this gap. With the HRADIO lab playout system, the consortium partners are able to locally set up a combined playout of real onAir DAB+ and IP streamed radio services, which includes correct DL+ and Slideshow data services for the programme elements played in the radio service (i.e. title and artist, genre ...). These services can be used by the project partners for the development of the HRADIO pilots and demonstrations, as well as for public demonstrations of the HRADIO system, where local receivable radio services do not provide the necessary level of data services and metadata.
TABLE OF CONTENTS

1. INTRODUCTION.................................................................................................................. 8
2. ARCHITECTURE .................................................................................................................. 9
3. INSTALLATION AND INFORMATION ABOUT THE CONTAINERS................................. 14
   3.1. General information........................................................................................................ 14
   3.2. Version using LibreTime ............................................................................................... 15
      3.2.1. Installation ............................................................................................................. 15
      3.2.2. Information about the containers .......................................................................... 16
   3.3. Version using MPD ........................................................................................................ 16
      3.3.1. Installation ............................................................................................................. 16
      3.3.2. Information about the containers .......................................................................... 17
4. FUTURE PLANS.................................................................................................................. 18
5. CONCLUSION....................................................................................................................... 19
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE 1: LAB PLAYOUT ARCHITECTURE</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIGURE 2: SCREENSHOT LIBRETONE UI.</td>
<td>10</td>
</tr>
<tr>
<td>FIGURE 3: SCREENSHOT YMPD MUSIC PLAYER FRONTEND</td>
<td>11</td>
</tr>
<tr>
<td>FIGURE 4: SCREENSHOT YMPD MUSIC PLAYER FRONTEND</td>
<td>13</td>
</tr>
</tbody>
</table>
## ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DAB</td>
<td>Digital Audio Broadcasting</td>
</tr>
<tr>
<td>DNS</td>
<td>Dynamic Name Server</td>
</tr>
<tr>
<td>EDI</td>
<td>Encapsulated Data Interface</td>
</tr>
<tr>
<td>ETI</td>
<td>Ensemble Transport Interface</td>
</tr>
<tr>
<td>EPG</td>
<td>Electronic Programme Guide</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>LTS</td>
<td>Long Term Support</td>
</tr>
<tr>
<td>ODR</td>
<td>Open Digital Radiotools</td>
</tr>
<tr>
<td>PAD</td>
<td>Program Associated Data</td>
</tr>
<tr>
<td>MPD</td>
<td>Music Player Deamon</td>
</tr>
<tr>
<td>rbb</td>
<td>Radio Berlin Brandenburg</td>
</tr>
<tr>
<td>RDS</td>
<td>Radio Data System</td>
</tr>
<tr>
<td>SSF</td>
<td>Second Screen Framework</td>
</tr>
<tr>
<td>SI</td>
<td>Service Information</td>
</tr>
<tr>
<td>SPI</td>
<td>Service and Programme Information</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>VRT</td>
<td>Vlaamse Radio- en Televisieomroeporganisatie</td>
</tr>
<tr>
<td>VUB</td>
<td>Vrije universiteit Brussel</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

As the different developments of HRADIO (API, libraries, Apps ...) are distributed amongst the project partners which spread over four European countries, a common testing and prototyping platform is necessary. While the distribution and simultaneous development of software today is a common task and the necessary devices are available, the simple sharing of a broadcast signal might become complicated.

In order to enable the different development partners to reproduce in their lab environment test radio signals within a real live scenario, a simple lab playout platform shall be developed in HRADIO task 3.4.

Furthermore, it is very likely, that the broadcast partners (i.e. rbb and VRT) will not have all the necessary signals, metadata and content items in their regular services. Therefore, the Lab Playout system additionally can be used to set up demonstration services with the full support of DAB data services.

This deliverable reports on the development of such Playout system. The next Section discusses its architecture. Section 3 then explains the concrete steps needed for doing an installation of the system.
2. ARCHITECTURE

The whole HRADIO lab playout consists of different components running within docker\(^1\) containers. Docker-compose\(^2\) is used to manage the different containers.

The following figure show the architecture and the components of the lab playout system.

![Lab Playout Architecture Diagram](image)

**Figure 1: Lab Playout architecture**

The Lab Playout system for HRADIO consists of four main parts:

1. **Music Playout**
   The Music Playout takes care to provide an audio stream and metadata (Title/Artist, Cover Art, ...) for the playout system. HRADIO uses two music playout systems to satisfy different requirements. The first one to have a

\(\text{https://www.docker.com/}\)

\(\text{https://docs.docker.com/compose/}\)
nearly full feature Radio Station, the other to create a quick and easy to use playout.

a. Libre Time\(^3\) : An open source radio software for scheduling and station management. With LibreTime, it is possible to upload audio files, create shows/schedules and stream audio via Icecast. The audio stream can be used as an input for the Opendigitalradio tools. LibreTime can be accessed and configured via a WebGUI.

Using LibreTime, HRADIO can provide a nearly full featured Radio Station. The operator can setup different users, who can plan the schedule of a radio station consisting of “Shows” with music and other audio content (e.g spoken Podcasts). Live insertion of audio is possible, also the access to cloud based audio sources such as “Soundcloud”\(^4\)

![Figure 2: Screenshot LibreTime UI.](image)

b. MPD\(^5\) : The second music playout system is build around the Music Player Daemon (MPD). MPD is an open source server-side music

---

\(^3\)https://github.com/LibreTime/
\(^4\)https://soundcloud.com
\(^5\)https://www.musicpd.org/
player that can be extended via different plugins and tools and can be controlled over the network. In addition to MPD we added some other tools:

i. ympd\textsuperscript{6}: A Web GUI for MPD
ii. ashuffle\textsuperscript{7}: Automatically adds songs to the queue to keep MPD playing

![Screenshot YMPD Music player frontend.](image)

**Figure 3:** Screenshot YMPD Music player frontend.

2. PAD Inserter: As each Music playout provides different ways to access the metadata of an audio track, two different PAD-Inserters were created. These applications get the metadata (Song, Artist, Cover Art, ...) from the corresponding Music playout, converts it into the appropriate format and sends it to the Opendigitalradio tools. The PAD encoder inside the opendigitalradio tools will use this data to create the slideshow and dynamic label for DAB.

---

\textsuperscript{6} https://www.ympd.org/
\textsuperscript{7} https://github.com/joshkunz/ashuffle
3. Opendigitalradio (ODR\textsuperscript{8}) tools: Open source tools to create a complete DAB transmission chain including audio and Program-Associated Data (PAD) encoder, multiplexer and modulator. The modulator can be connected via USB to a SDR-transmitter like the USPR Platform from Ettus\textsuperscript{9}.

4. ODR-Control (In Development): The Opendigitalradio tools can only be configured via different configuration files. As a result, it can be hard to make changes on the fly without a lot of knowledge about DAB. The ODR-Control part will provide an easy to use Web GUI to change the essential settings (DAB Channel, Source, Output, Ensemble and Service Data like Names, IDs, ...) of the Opendigitalradio tools. There will also be an option to upload a previous recorded Ensemble Transport Interface (ETI) file and put it directly on air. ETI is a standardised exchange format for DAB Ensemble recordings. Initially specified for the transport of DAB ensembles between distant multiplexer and modulator installations, it evolved also to a common file exchange format.

5. EDI-Splitter: The DAB–IP splitter is a server component, which re-multiplexes the full EDI ensemble multiplex into single service EDI streams, which can be delivered to HRADIO clients over HTTP/HTTPS connections.

The playout system is published using a docker-compose file. Using docker and docker-compose, enables an out-of-the-box deployment of the whole playout using just one command, making it to distribute and nearly hardware independent. Additionally, for Docker based deployments a lot of tools for deployment, management and monitoring exists and enable a nearly 100% command line free operation. See below the portainer.io\textsuperscript{10} GUI for management and monitoring of the HRADIO lab playout.

\begin{itemize}
\item[8] https://github.com/Opendigitalradio/
\item[9] https://www.ettus.com/
\item[10] https://www.portainer.io
\end{itemize}
Figure 4: Screenshot YMPD Music player frontend.
3. INSTALLATION AND INFORMATION ABOUT THE CONTAINERS

3.1. GENERAL INFORMATION

For a clean installation of the HRADIO Lab Playout system, all the necessary resources are either available as free software components or are stored on the HRADIO GIT repositories under:

https://gitlab.irt.de/hradio/platforms/omri-lab-playout

The GIT repository currently contains two different branches. One contains the resources for the LibreTime based system (Branch: airtime), the other is the MPD based version (Branch: master).

The installation has been tested on various Linux installations running Ubuntu 18.04(LTS) and Ubuntu 18.10. At least 8GB of RAM and a i5 or i7 4 core CPU are recommended.

For the terrestrial output of the modulated DAB+ signal the following hardware is supported:

- LimeSDR\textsuperscript{11} mini (default)
- ETTUS\textsuperscript{12} B100, B200, USRP1
- HackRF\textsuperscript{13} (not tested)

As antenna we recommend:

- Antennentechnik-Bad-Blankenburg\textsuperscript{14} (good high quality build)

\textsuperscript{11} https://www.crowdsupply.com/lime-micro/limesdr-mini
\textsuperscript{12} https://www.ettus.com/
\textsuperscript{13} https://greatscottgadgets.com/hackrf/
\textsuperscript{14} https://www.antennensysteme.de/produkte/produkte/detailansicht/news/4929-01-stationaere-antenne-dab-biii-l-band/
3.2. VERSION USING LIBRETIME

3.2.1. Installation

- install docker + docker-compose

- download the repository onto the machine using the correct tag

- To build all containers, run `docker-compose build`. Building the containers will take some time

- create and start all containers using `docker-compose up -d`

- open the Setup-Page of LibreTime (should be on localhost, port 80 <http://localhost>). It should be possible to use the default values for the setup

- after the Setup, some errors showing that there are unstarted services will be displayed. Refresh the page a few times and the errors will be gone

- Login to LibreTime using admin:admin, upload some music and create a show/schedule

- bash into libretime to get the API-Key from `/etc/airtime/airtime.conf` via `docker exec -it libretime /bin/bash` and add it to the docker-compose.yml Line 35 as `HRADIO_SCHEDULEURL=http://libretime/api/schedule/api_key/[KEY]`

- restart libretimetoodr (the PAD Inserter) via `docker-compose restart libretimetoodr`
### 3.2.2. Information about the containers

- **libretime:**
  - Port 80: WebGui
  - Port 5672: RabbitMQ to connect libretimetoodr
  - Port 8000: Icecast
  - All services started automatically via the scripts inside `libretime/supervisor/

- **odr:**
  - All services started automatically via the scripts inside `odr/supervisor/
  - All connected USB devices are mounted on start (`/dev/bus/usb/`)
  - Settings for audio, pad, multiplex and modulation can be found in `odr/dab/
  - The slideshow and dynamic label can be found in `odr/dab/

- **libretimetoodr:**
  - Writes all slides and dynamic label into `odr/dab/`

### 3.3. VERSION USING MPD

#### 3.3.1. Installation

- Install docker + docker-compose
- Download the repository onto the machine using the correct tag
- To build all containers, run `docker-compose build`. Building the containers will take some time
- Store some mp3 files in music/. MPD will use these files
- Create and start all containers using `docker-compose up -d`
- Open a browser to connect to ympd (usually port 80) and enable “consume” (track gets deleted from the queue after played)
3.3.2. Information about the containers

- **mpd:**
  - Port 6600: for the mpd control protocol
  - Port 8000: for Icecast
  - config file is `mpd/mpd.conf`

- **ympd:**
  - Port 80: For the WebGui

- **odr:**
  - All services started automatically via the scripts inside `odr/supervisor/`
  - All connected USB devices are mounted on start `(/dev/bus/usb/)`
  - settings for audio, pad, multiplex and modulation can be found in `odr/dab/`
  - The slideshow and dynamic label can be found in `odr/dab/`

- **mpd-pad (PAD Inserter):**
  - configured via `mpd-pad/appsettings.json`
  - writes all slides and dynamic label into `odr/dab/`

Useful commands for docker-compose:

- `'docker-compose ps'`: shows the current running containers and their status/open ports
- `'docker-compose start|stop|restart [containername]'`: starts, stops or restarts the corresponding container
- `'docker-compose logs [containername]'`: shows the logs of the corresponding container. You can use the switches `-f` to follow and display upcoming logs and/or `-t` to display timestamps
4. FUTURE PLANS

Currently the Opendigitalradio tools can only be configured via various files and have to be restarted if something got changed. In the future there should be a convenient way for all partners to change the most needed settings (Frequency, ensemble name, audio encoding, ...). There will also be a way to broadcast a pre-recorded Ensemble. These features will be combined inside the ODR-Control component. A Web-Gui will also be developed that provides each partner an easy way to change the settings of the DAB output to his needs.
5. CONCLUSION

This deliverable covered the work of T3.4 in WP3. The main goal of this task is to deliver a fully featured DAB+/DAB-IP playout system for development and demonstration purposes. A secondary goal was to enable also non-technical users to deploy the system without all the complicated set up of compilers, libraries and other dependencies needed to operate a opendigitalradio.org tool chain. The current status of the task has fulfilled this to goals completely. Basically 3 commands on a Linux terminal finish with a running DAB+/DAB-IP radio service, receivable with every standard DAB+ radio device. The radio service contains all the necessary data services and metadata, even for more complicated HRADIO use cases such as time shift with programme item related access. Future enhancements will cover more UI developments, in order to make configuration possible through web-based UI instead of editing configuration files.