This deliverable includes a state-of-the-art assessment of available technologies and standards in the hybrid radio field. Following this, a comprehensive overview over the HRADIO technical architecture is given.
### Basic Information

<table>
<thead>
<tr>
<th>Work package</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Due date</td>
<td>30/09/2019</td>
</tr>
<tr>
<td>Submission date</td>
<td>30/04/2020</td>
</tr>
<tr>
<td>Deliverable lead</td>
<td>IRT</td>
</tr>
<tr>
<td>Version</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Authors**

Alexander Erk (IRT), Markus Friedrich (LMU), Andre Ebert (LMU), Maximilian Knop (Konsole), Klaas Baert (VRT), Leo Andrews (Radioplayer)

**Reviewers**

Dominique Hazael-Massieux (W3C), Francois Daoust (W3C), Maximilian Knop (Konsole)

### 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>09/05/2018</td>
<td>Initial version</td>
<td>Alexander Erk (IRT)</td>
</tr>
<tr>
<td>V0.2</td>
<td>26/06/2018</td>
<td>Updates state-of-the-art</td>
<td>Alexander Erk (IRT), Markus Friedrich (LMU), Andre Ebert (LMU), Maximilian Knop (Konsole), Klaas Baert (VRT), Leo Andrews (Radioplayer)</td>
</tr>
<tr>
<td>V0.9</td>
<td>30/07/2018</td>
<td>Final version for review</td>
<td>Alexander Erk (IRT), Markus Friedrich (LMU)</td>
</tr>
<tr>
<td>V1.0</td>
<td>01/08/2018</td>
<td>Internal and coordinator review</td>
<td>Dominique Hazael-Massieux (W3C), Maximilian Knop (Konsole), Simon Delaere (imec)</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Description</td>
<td>Reviewers</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>--------------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>V1.5</td>
<td>18/02/2020</td>
<td>Draft 2\textsuperscript{nd} version for review</td>
<td>Aleander Erk (IRT), Fabian Sattler (IRT), Markus Friedrich (LMU)</td>
</tr>
<tr>
<td>V1.9</td>
<td>03/03/2020</td>
<td>Final version from internal review</td>
<td>Francois Daoust (W3C), Maximilian Knoop (konsole)</td>
</tr>
<tr>
<td>V2.0</td>
<td>30/03/2020</td>
<td>Coordinator review</td>
<td>Simon Delaere (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>Nature of the deliverable:</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dissemination Level</strong></td>
<td></td>
</tr>
<tr>
<td>PU</td>
<td>Public, fully open, e.g. web</td>
</tr>
<tr>
<td>CL</td>
<td>Classified, information as referred to in Commission Decision 2001/844/EC</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential to HRadio project and Commission Services</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

This is the final version of the overall HRADIO System architecture. This document includes an extensive state of the art assessment of available technologies and standards in the field of hybrid radio. The document also analyses the derived system requirements from HRADIO deliverable D2.4, assigns them to WP3 tasks and provides a mapping of these requirements to the actual state of the art technologies. Finally, a description of the overall system architecture and its components is given in section 3.

The state-of-the-art analysis and the mapping of the system requirements to the related state of the art technologies showed, that for most of the system requirements technological solutions exist and are available. Especially the current state of the OMRI API supports many radio centric requirements.

The HRADIO technical work is focusing on three basic blocks. The Client, the metadata platform and the playout. Section 3 gives an overview of this work and the solutions which have been developed in WP3.

In the first iteration of this document, the HRADIO architecture was already quite mature. However, this final version describes a much more detailed client library architecture, especially for the web application part. It provides a comprehensive overview of the lab playout including the reasoning of the DAB-over-IP developments and a complete architecture of all components of the HRADIO content and Metadata platform.

Finally, a newly added section 4 provides an overview, on how all these components work together in the HRADIO pilot applications.
# TABLE OF CONTENTS

## 1. STATE OF THE ART ANALYSIS

1.1. Broadcast Systems

1.1.1. DAB+

1.1.2. FM 18

1.2. IP delivered Audio

1.2.1. Shoutcast/Icecast

1.2.2. Adaptive streaming systems

1.3. Hybrid Radio Solutions

1.3.1. RadioDNS

1.3.2. Device Pairing

1.4. Radio Clients

1.4.1. LG Stylus

1.4.2. Konsole Player platform

1.4.3. RadioPlayer

<table>
<thead>
<tr>
<th>TABLE D3.1: HRADIO overall system architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Radio Solutions</td>
</tr>
<tr>
<td>RadioDNS</td>
</tr>
<tr>
<td>RadioTAG</td>
</tr>
<tr>
<td>EBU CPA Cross Platform Authentication</td>
</tr>
<tr>
<td>IRT Second Screen Framework</td>
</tr>
<tr>
<td>RadioPlayer on the LG Stylus 2</td>
</tr>
<tr>
<td>Radioplayer in the car</td>
</tr>
</tbody>
</table>

Page 6 of 75
1.4.4. IRT DABerry ........................................................................................................................................32
   Hardware and Software of DABerry ........................................................................................................32
   Skipping/Time shift ..................................................................................................................................36
   Recording ..................................................................................................................................................37
   RadioWEB ..................................................................................................................................................37
1.5. Metadata Platforms ................................................................................................................................38
1.6. Distributed Database Technologies ......................................................................................................42
1.6.1. Distributed Hash Tables (DHTs) ......................................................................................................42
1.6.2. Distributed NoSQL Document Stores ............................................................................................43

2. DERIVED TECHNICAL REQUIREMENTS AND GAP ANALYSIS ..................................................46
2.1. Derived technical requirements ........................................................................................................46
2.2. Mapping of requirements to actual state of the art technology ..........................................................54
2.2.1. T3.2 requirements .........................................................................................................................54
2.2.2. T3.3 requirements .........................................................................................................................55
2.2.3. T3.4 requirements .........................................................................................................................57

3. FINAL HRADIO SYSTEM ARCHITECTURE .......................................................................................58
3.1. Client Libraries T3.3 ..........................................................................................................................60
3.1.1. Android ..........................................................................................................................................61
3.1.2. iOS ..................................................................................................................................................62
3.1.3. HTML/Browser ............................................................................................................................62
3.2. Lab Playout T3.4 ..................................................................................................................................63
3.3. HRADIO Content and Metadata Platform ........................................................................................67

4. THE BIG PICTURE .................................................................................................................................72

5. CONCLUSION ........................................................................................................................................74

REFERENCES ...............................................................................................................................................75
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Constant flow of message items in DL/DL+</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Constant flow of slideshow images in DAB-SLS</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Categorized DAB-SLS with local device cache</td>
<td>16</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Link URL to more related information via “click-through-url” in categorized slideshow</td>
<td>17</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Different DAB-SPI device implementations</td>
<td>17</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Common Shoutcast related brands</td>
<td>20</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Adapation of quality during playback</td>
<td>22</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Apple HLS streaming</td>
<td>23</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Dash streaming</td>
<td>23</td>
</tr>
<tr>
<td>Figure 10</td>
<td>General Radiodns lookup</td>
<td>25</td>
</tr>
<tr>
<td>Figure 11</td>
<td>LG Stylus DAB+ smartphone</td>
<td>29</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Daberry Hardware</td>
<td>33</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Daberry HTML GUI with DL+, slideshow and DAB-SPI</td>
<td>34</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Final Daberry prototype</td>
<td>35</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Buffer segmentation based on DL+</td>
<td>37</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Radioweb service from Antenne Bayernn embedded in Daberry UI</td>
<td>38</td>
</tr>
<tr>
<td>Figure 17</td>
<td>HRadio technical components overview</td>
<td>59</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Client libraries architecture</td>
<td>61</td>
</tr>
<tr>
<td>Figure 19</td>
<td>HRadio system architecture for web applications</td>
<td>63</td>
</tr>
<tr>
<td>Figure 20</td>
<td>DAB over IP</td>
<td>65</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Lab Playout architecture</td>
<td>66</td>
</tr>
<tr>
<td>Figure 22</td>
<td>1st Metadata platform</td>
<td>69</td>
</tr>
<tr>
<td>Figure 23</td>
<td>HRadio data model</td>
<td>71</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Overview of HRadio components used for pilot applications</td>
<td>72</td>
</tr>
</tbody>
</table>
LIST OF TABLES

TABLE 1: SYSTEM REQUIREMENTS ........................................................................................................ 47
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CPA</td>
<td>Cross Platform Authentication</td>
</tr>
<tr>
<td>DASH</td>
<td>Dynamic Adaptive Streaming over HTTP</td>
</tr>
<tr>
<td>DAB</td>
<td>Digital Audio Broadcasting</td>
</tr>
<tr>
<td>DAB-DL</td>
<td>DAB Dynamic Label</td>
</tr>
<tr>
<td>DAB-SPI</td>
<td>DAB Service and Programme Information</td>
</tr>
<tr>
<td>DAB-SLS</td>
<td>DAB Slideshow</td>
</tr>
<tr>
<td>DNS</td>
<td>Dynamic Name System</td>
</tr>
<tr>
<td>EPG</td>
<td>Electronic Programme Guide</td>
</tr>
<tr>
<td>FM</td>
<td>Frequency modulation</td>
</tr>
<tr>
<td>HbbTV</td>
<td>Hybrid broadband broadcast Television HTML Hypertext Markup Language</td>
</tr>
<tr>
<td>HLS</td>
<td>HTTP Live Streaming</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>IDAG</td>
<td>International DMB Advancement Group</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>PI</td>
<td>Programme Information</td>
</tr>
<tr>
<td>MOT</td>
<td>Multimedia Object Transfer</td>
</tr>
<tr>
<td>SI</td>
<td>Service Information</td>
</tr>
<tr>
<td>SPI</td>
<td>Service and Programme Information</td>
</tr>
<tr>
<td>STOMP</td>
<td>Simple Text Oriented Messaging Protocol</td>
</tr>
</tbody>
</table>
TCP  Transmission Control Protocol
RBB  Rundfunk Berlin-Brandenburg
VRT  Vlaamse Radio en Televisieomroeporganisatie
VUB  Vrije universiteit Brussel
XML  eXtensible Markup Language
1. STATE OF THE ART ANALYSIS

In the following section, the document will describe the state of the art of technology related to the HRADIO project. As broadcast systems, DAB and FM will be analysed not so much with a focus on the audio transmission but more with additional data services they enable as they are the most relevant to HRADIO. With regard to IP based audio services, the streaming technologies Shoutcast/Icecast as well as adaptive technologies such as HLS and DASH are described. HRADIO concentrates on radio service innovations based on the convergence of broadcast and IP delivered content, often abbreviated as hybrid radio; the hybrid radio specification from RadioDNS will therefore follow and will be complemented by technologies for device pairing (IRT SSF and EBU-CPA). After this technology overview, the actual work on hybrid radio clients will be investigated. In the last section of this analysis, current metadata and distributed database platform technology is investigated, as it relates to the development of the distributed metadata and communication platform for HRADIO.

1.1. BROADCAST SYSTEMS

1.1.1. DAB+

Between 1987 and 2000, a transmission standard called Digital Audio Broadcasting (DAB) was developed in the Eureka 147 project of the European Union (EU). DAB is suitable for the frequency range from 30 MHz to 3 GHz and therefore also includes the possibility of distribution via cable and satellite. DAB has been available in Germany since 1995. However, coverage and supply vary from region to region. It was planned that DAB would replace the analogue transmission in the VHF band.

For the listener, DAB/DAB+ offers a number of additional services and functions in addition to digital radio content. All in all, this results in a clearly recognisable added value compared to analogue distribution via VHF. The advantages for the listener are:
D3.1: HRADIO overall system architecture

➔ More content
➔ Better quality
➔ New functions and services (images, TPEG, EPG, etc.)

DAB/DAB+ Dynamic Label/Dynamic Label+ (DL/DL+)

With Dynamic Label (DL)/Dynamic Label+ (DL+), short text messages are transmitted in the DAB/DAB+ signal. Almost all devices can represent this standard. The text messages can be “tagged” and summarized. From a DL/DL+ message such as “You hear: Start Me Up from the Rolling Stones”, the title of the music and the name of the artist can be extracted.
**DAB/DAB+ SlideShow/Categorised SlideShow**

With the DAB/DAB+ slideshow, images (in PNG and JPEG format) can be transmitted together with audio content. The "slides" provide additional information accompanying the program in the form of images that devices can display.
DAB/DAB+ Categorised Slideshow is an extension of the DAB/DAB+ slideshow and overcomes some limitations of the slideshow. Slideshows can now be stored in the DAB receiver and use links to websites. The DAB user can then view the slides or access additional offers on the Internet at a later point in time.
The following picture shows the possible linking of slide images with additional background information on the service provider’s website.
D3.1: HRADIO overall system architecture

**Figure 4:** Link URL to more related information via “click-through-url” in categorized slideshow.

**DAB–SPI**

The DAB–SPI informs about the future schedule of the program. The specification itself defines an XML structure based on TV anytime, allowing a rich set of metadata (images, descriptions, links).

The DAB–SPI contains service information about the accessible services, program information and logos.

**Figure 5:** Different DAB–SPI device implementations

**TPEG**

TPEG (Transport Protocol Experts Group) technology was motivated by the desire to develop a modern multimodal TTI (Traffic and Travel Information)
data protocol for delivering content to the end-user. The predecessor technology RDS-TMC had significant limitations, is essentially limited to road events and need a synchronized location database for the message geo-referencing. TPEG overcomes this limitation by the introduction of various location referencing methods, which can deliver on-the-fly location information, so that client devices do not need a location database.

Furthermore, TPEG technology is designed to facilitate many more applications covering many other aspects of the TTI domain, such as TFP (Traffic Flow and Prediction) for providing clients with up-to-date and upcoming traffic state information, PKI (Parking Information), WEA (Weather Information) etc. Currently the whole TPEG framework consists of ten applications and five location referencing methods. TPEG also permits the use of encryption mechanisms, if required by an application. The TPEG format is bearer independent: it is suitable both for the broadcast delivery and for the connected services. The umbrella organization for the TPEG standard is TISA (Traveller Information Services Association), which is focused on implementation and deployment issues, specifications support and maintenance/development issues. Countries where the TPEG services are actually operational or planned can be seen in the TISA interactive map¹.

1.1.2. FM

VHF is an analog transmission method. The modulation signal of FM consists of a part for the mono signal, a stereo signal part (difference signal), a pilot tone at 19 kHz and the RDS carrier, which is phase modulated at 57 kHz. RDS is the radio data system in which additional information such as station identification, station name, text information or traffic information (TMC) is transmitted.

In general, besides the audio service, data services such as radiotext, station identification, program type, traffic announcements and in some cases RadioText+ are supported by using the RDS system. The transmission of traffic information via TMC plays an important role in the transmission of safety-relevant traffic data. Most of the navigation solutions, whether internal or external, are TMC capable and the public broadcasters transmit this data.

¹ http://tisa.org/technologies/tpegtm-services-map/
RDS services in the 57 kHz subcarrier can utilize a data bandwidth of approx. 1.187 kBit/s.

Specified services are:

- **AF** (alternative frequencies list) is used for automatic frequency selection in case that the first signal becomes unavailable.
- **CT** (clock time and date) provides clock synchronisation up to an accuracy of 100ms.
- **EON** (enhanced other networks information) monitor other networks/services.
- **PI** (programme identification) provides a unique code for station identification. This feature is used to generate the unique RadioDNS FQDN for the lookup of hybrid services for FM stations.
- **PS** (programme service name) provides 8 characters for display of the station name.

### 1.2. IP DELIVERED AUDIO

The standards HTML5 and MPEG-4/H.264/H.265/AAC/MP3 are appropriate for cross-device distribution of content over the open internet. Suitable combinations of container formats, transport protocols and media coding schemes need to ensure interoperability and maintain perceptive quality. IP-based delivery mechanisms need to support a multimedia transport based on the consumers’ internet connection and their individual network access. Special requirements emerge for object based Next Generation Audio (NGA) and the seamless switching of modern audio coding schemes such as HE-AAC, Opus or AC4. While transmission can take the form of downloading, pseudo streaming (Shoutcast/Icecast) or live streaming through different protocols (HTTP, QUIC, WebRTC, SRT), bottlenecks need to be compensated by load balancing and caching in Content Delivery Network-Providers (CDN) or via client-based transport mechanisms like Adaptive Streaming.
1.2.1. Shoutcast/Icecast

Shoutcast and Icecast are the two main implementations of streaming via theICY protocol. This protocol is built on HTTP and allows the transport of an encoded audio signal as a bitstream, from a server to a client. It was developed in the late 1990’s by Nullsoft, the company that developed Winamp, a formerly popular software to play (streaming) audio.

Both Shoutcast and Winamp are currently owned by Radionomy, a company that operates an ad-supported platform for streaming radio under the same name, and an advertisement platform called Targetsport. This setup allows radio broadcasters to set up an online station for free, and if their listenership is significant enough, take a share of the revenue. The Shoutcast DNAS (Distributed Network Audio Server) can be run on self-owned dedicated hardware, or it can be hosted by Radionomy. In the latter case enrollment in the monetization program is mandatory. Monetization on the Radionomy platform is enabled in the US, France, Germany, UK, Belgium, Spain, Switzerland, Argentina, Canada, Ireland, Mexico and the Netherlands. The current version of the Shoutcast DNAS is v2.5. This is a version that supports MP3 and AAC encoded streams. The server produces a URL endpoint that can be accessed directly, or that can be relayed to a content delivery network. This works completely transparent because of the HTTP basis of ICY.

Icecast is an open source implementation of a server that exposes audio streams via the ICY protocol. Icecast currently supports Ogg Vorbis and MP3 encoded audio streams. The stable release of Icecast at the time of writing is version 2.4.3, with a new 2.5 version in open beta. Since this is an open source project there is no larger framework nor a company in place that enable a revenue model. However, there is a large community of contributors that writes code for the development of Icecast, which makes it a trustworthy system that simply does what is expected from it.
Most audio sources that are capable of connecting directly to streaming servers support both Shoutcast and Icecast, because they both use the same underlying protocol. There are some differences, like the absence of the "mountpoint" concept in older Shoutcast servers, but these differences are mostly tackled in the UI of these audio sources.

Both Shoutcast and Icecast offer a Yellow Pages (YP) system, a listing of online stations that facilitates discovery and search of stations’ streaming URLs. Currently it is impossible to list Shoutcast stations in the Icecast YP and vice-versa.

1.2.2. Adaptive streaming systems

With adaptive delivery using HLS or MPEG-DASH, a wide range of devices can access content over the open internet without annoying the users with rebuffering or stalling playback even when the internet access is limited or unstable. Adaptive Streaming was introduced to overcome bottlenecks in the IP-based delivery of multimedia content over the open internet. Bandwidth variations will cause the player to seamlessly switch to another suitable representation of the same content through a regular HTTP web server, without invoking rebuffering, stalling or stopping the playback.

Adaptive Streaming Services can support both video and audio including subtitles or audio-only signals e.g. for a radio channel. Various audio tracks e.g. exposing different languages, clean audio or alternate viewports can be carried in a single media presentation. Bitrate-adaptive playback typically is needed to deliver high bitrate video over limited bandwidth connections without quality of service. Nevertheless radio-services based on high efficiency audio codecs can benefit from adaptive streaming even when the media bitrate is low compared to UHD or 360 video. Compared to legacy shoutcast/icecast audio transport, adaptive delivery can ensure gapless audio playback through cellular handover in mobile environments down to low UMTS or even EDGE network speeds. Furthermore multiple channel- configurations from Mono up to 5.1 or 7.1 can be offered through a single manifest file.
Figure 7: Adaption of quality during playback

A reasonable trade-off between coding efficiency, real-time delay, caching performance and seeking responsiveness can be obtained using segment-durations from 4 Seconds to 6 Seconds. The overall end-to-end latency of a complete transmission change over the open internet shall not exceed 30 seconds for adaptive streaming, while optimized Adaptive Low Latency Streaming can be provided at less than 5 Seconds. Smartphones, tablets, PC and HbbTV are the main targeted client platforms in Germany. Apple iOS and Android clients are provided with traditional HLS Adaptive Streaming using MPEG2-TS segments; on PCs, either legacy Progressive Download (PD) in an MP4-container, MPEG-DASH carrying ISO/MBFF-segments supporting both mutliply audio and EBU-TT-D subtitles can be delivered, depending on users browser and OS-configuration with HTML5/MSE. Nearly all Smart TVs currently sold in European countries are equipped with HbbTV, either in Version 1.0 requiring Progressive Download, or in Version 1.5 and 2.0 adding MPEG-DASH Media Transport as defined in DVB-DASH following the HbbTV-Standards ETSI TS 102 796 in its versions ETSI 1.1.1 (HbbTV 1.0), ETSI 1.2.1 (HbbTV 1.5) and ETSI 1.4.1 (HbbTV 2.0.1).

HLS

HLS is a transport mechanism for transmitting multimedia data over HTTP. It defines a playlist format called M3U8 and a media segment format based on MPEG-2 TS or ISO/MBFF MPEG-4 Fragmented starting from revision 20 (2016). HLS supports the adaptive delivery of alternate versions (e.g. bitrates, languages) of media data. It is suitable for both live and on-demand
playback. The media stream is separated into Media Segments, which last from 2 to 10 seconds.

**DASH**

MPEG-DASH (Dynamic Adaptive Streaming over HTTP, ISO/IEC 23009–1) is a vendor independent, international standard ratified by MPEG and ISO. Previous adaptive streaming technologies – such as Apple HLS, Microsoft Smooth Streaming or Adobe HDS had been released with limited support of company-independent streaming servers and playback clients. The standardization bodies started a harmonization process, resulting in the publication of MPEG-DASH in 2012.

Benefits of MPEG-DASH are the reduction of startup delays, buffering or stalls during the playback with a continued adaptation to the bandwidth situation of the client. DASH introduces a client-based streaming logic enabling highest scalability and flexibility using cost-effective HTTP-based CDNs, proxies and caches. It naturally bypasses NATs and firewalls since it is based on HTTP. MPEG-DASH playback is enabled by HTML5 Media Source Extensions (MSE) via the HTML5 video and audio tags. MPEG-DASH in general only defines the Media Presentation Description (MPD) as XML and the Segmentation of the Media Streams.

It is codec-agnostic and the client behaviour is out of scope to support a wide range of applications including audio-only streaming and profiles for...
live- and on demand-services. Various descriptors for the MPD are defined e.g. for accessibility, parental rating, multiple language support, subtitles and audio channel configurations. Separate program items can be comprised by exposing multiple periods within a MPD. Program related events can be triggered through an intermediate refresh of the MPD.

BBC Radio 3 and BBC R&D launched a classical radio network in 2017 with audio delivered directly to the web browser with completely lossless compression using MPEG DASH and FLAC compression\(^2\). HTML5 and the Media Source Extensions ensure a bit-perfect representation of Radio 3’s live output.

1.3. HYBRID RADIO SOLUTIONS

1.3.1. RadioDNS

RadioDNS is an open set of specifications for the implementation of hybrid radio services. RadioDNS focuses on broadcasting, based on the perspective that broadcasting is still the best medium to transmit a radio programme efficiently and reliably for many people. IP, on the other hand, is the perfect choice for the individual enrichment with multimedia content, e.g. pictures, additional texts and interactive offers. RadioDNS aims at combining radio and internet to make radio better.

\(^2\) [https://www.bbc.co.uk/rd/blog/2017-04-radio-3-high-quality-flac-dash](https://www.bbc.co.uk/rd/blog/2017-04-radio-3-high-quality-flac-dash)
A fundamental problem of hybrid broadcasting services is finding a service on the Internet based on the currently received broadcasting service. This includes various broadcasting systems as well as backwards compatibility and easy commissioning.

RadioDNS solves this problem by utilizing the standardised and well known DNS system. The picture above shows the general sequence of DNS requests and the corresponding requirements for the search of a RadioDNS service for a radio program.

(1) A host name is formed from the reception parameters of the tuned radio program according to a scheme defined in RadioDNS. This host name will be resolved to a CNAME record (2) via the common DNS infrastructure. This CNAME is the host to which further service record requests are made. According to the desired RadioDNS service, the client formulates one or more service record requests (3) which, if successful, lead to the respective web server for this particular service.

Currently the RadioDNS group has delivered three service specifications: RadioVIS, RadioEPG and RadioTAG. In 2014 RadioDNS together with WorldDAB decided to harmonize the existing WorldDAB Slideshow and EPG ETSI specification with their counterparts RadioVIS and RadioEPG into a set of two common specifications:
RadioTAG

Radio tends to be used as an accompanying media, which can be consumed under a large variety of different usage scenarios. Although the responsible radio people like to encourage listeners to further engage with the radio programme, often this engagement is difficult to implement (e.g. call in via phone) and difficult to perform (e.g. in car). RadioTAG tries to overcome this problem, by proposing a simple interaction mechanism such as simply pressing a button, e.g. in the case a listener wants to express their interest in a song. When the RadioTAG function on a device (either pressing a button or issuing a voice command) is activated, the client sends a request to the RadioTAG server (discovered over the DNS queries) specifying time and service. The server responds by sending back relevant metadata and may store this and the clients data for reuse.

The format of the data in server responses uses the XML-based Atom Syndication Format [ATOM] completed with elements in the RadioTAG Namespace.

Example:

```xml
<?xml version="1.0"?>
  <title>Meet David Sedaris</title>
  <link href="http://radiotag.bbc.co.uk"/>
  <link href="http://radiotag.bbc.co.uk" rel="self"/>
  <updated>2011-08-01T11:38:38+01:00</updated>
  <author>
```

3 RFC 4287: “The Atom Syndication Format”
BBC, together with FrontierSillicon and Revo (manufacturer of domestic radio devices), implemented a RadioTAG testbed. End users were able to “pair” their radio device with their BBC Web account (via EBU CPA). Once this pairing was established, end users could tag a piece of content by simply pressing a RadioTAG button on the device’s UI.4

1.3.2. Device Pairing

**EBU CPA Cross Platform Authentication**

CPA is a protocol specifically designed for devices with limited input and display capabilities that are not addressed by existing standards and to cater for companies that share a back-end authorization provider for managing identities but implement services separately.

---

4 [https://www.bbc.co.uk/rd/projects/radiotag](https://www.bbc.co.uk/rd/projects/radiotag)
The CPA protocol defines a clear separation of responsibilities between the service provider and the authorization provider. This enables the protocol to be applied in a variety of business configurations typical of the broadcast industry. For example, the CPA protocol can be used in the following scenarios:

→ the service provider and the authorization provider are both managed by the same company;

→ an umbrella company manages the authorization provider (so centralizing identity management) but keeps its

For further information regarding the EBU-CPA specification and its actual state please refer to the EBU project page and the actual version of the specification.

IRT Second Screen Framework

The Second Screen Framework (SSF) provides web applications which are running on a TV with all the crucial functionalities to establish a persistent bi-directional communication path to a web application running in the browser of any second-screen device. This includes the possibility to launch applications from one TV on the second screen. All functionalities are provided via a standard lightweight JavaScript API and can thus be easily integrated into any web application. The API enables a full duplex communication between web applications running in the browser of connected TV devices and second-screen devices. For set-up the device pairing, ITR-SSF uses a QR-Code based mechanism for device discovery. Once paired, it provides a persistent device connection – devices once coupled remain associated and enables an automatic launch of applications on the second-screen.

1.4. RADIO CLIENTS

1.4.1. LG Stylus

LG Stylus 2 was the first (and only) modern smartphone to feature a DAB tuner which was launched in Spring 2016. The Stylus 2 supported DAB+, which can decode and play both DAB and DAB+. 
LG also provided an Android SDK (called LG DMB SDK) for developers to access DAB tuner from within mobile apps. DMB is a digital radio transmission technology based on Digital Audio Broadcasting (DAB) for sending multimedia. DMB was developed in South Korea by importing DAB technology. The world’s first DMB broadcast service started in South Korea in 2005, and achieved International Standard status in 2007.

LG DMB SDK provides following key features:

- Listening to DAB broadcast: Searching DAB broadcast channels that only support voice, and listening to the desired channel
- Watching to DMB broadcast: Searching DMB broadcast channels that support voice and video, and watching the desired channel


Prior to the LG Stylus 2, Virgin launched the Mobile Lobster 700TV in 2006, which had DAB. The Nokia N8 and other models also provided DAB support via a Nokia CU-17A digital radio headset, which launched in 2011 and connected through the USB port.
The newer version of LG Stylus, LG Stylus 3, does not have a DAB tuner.

1.4.2. Konsole Player platform

Konsole Labs has been providing native apps for iOS, Android and Windows Phone operating systems for 7 years. All apps are natively developed so that problems and inefficiencies of Cross-Platform tools can be avoided.

The app uses the self-developed base of the current Konsole radio player (Version 5.1, December 2017).

In recent years, a large range of functions has emerged, which can be put together by the radio stations individually in their respective app. The following views have been developed for integration:

- Playlist
- Favourites function
- Upload Tool
- Survey
- Podcast Player
- DAB + reception (only on Android and on the LG Stylus 2)
- News (single list)
- Alarm clock

The last two points are services that have developed over the years as independent products: Konsole Labs uses their own push notification solution (www.Breaking-Push.com for SWR Aktuell, DLF24, Deutsche Welle), which does not use third-party software and stores data inside Germany.

For SWR Aktuell and Antenne Bayern Konsole Labs operates a weather warnings via push.

For data transfer, the apps use their own protocol via SSL which is currently available in the second version. Content is managed by the respective
editors via their own CMS, which provides the app with the configuration settings and content.

Key aspects in the last app version were the construction of a native podcast player, an improved audio player, use of a clear and classic UI and update the data transfer protocol to a new version level. With this protocol, even without UMTS, Konsole Labs can ensure that new content is transferred to apps faster than with normal JSON interfaces.

Apps that use this version:

- Radio SAW (iOS and Android)
- Radio Paloma (iOS and Android)
- Rockland FM (iOS and Android)

**1.4.3. RadioPlayer**

**Radioplayer on the LG Stylus 2**

The Radioplayer app is a streaming aggregator app which carries the services of hundreds of broadcast and online radio stations in the market within which it has been released. In first instance, the app is playing IP streams, however Radioplayer’s work with the LG Stylus 2 enhanced this: since this device has a DAB chip built into it, Radioplayer adapted the Radioplayer app to read bearers from RadioDNS and was successful in achieving true service following across platforms. This means a listener could start listening to a station using the Radioplayer app at work on the WiFi connection and, when leaving the office and getting in their car, switch to DAB seamlessly instead of consuming mobile data. A video of this can be seen at [https://www.youtube.com/watch?v=wj_nMRe_LPc](https://www.youtube.com/watch?v=wj_nMRe_LPc)

**Radioplayer in the car**

Radioplayer conducted an R&D project in 2015-2017 to build an aftermarket in car hybrid radio. This used multiple tuners, plus an IP connection provided by the user’s smartphone in order to provide service following across DAB, FM and IP. For the broadcast spectrum, a pair of tuners was used for each band. In this configuration, one FM tuner was used to continually scan the band for stations whilst another FM tuner was used to play the selected station.
Likewise, one DAB tuner performed a continual band scan whilst the other played the station.

True service following based on RadioDNS and Radioplayer’s proprietary database of bearer information meant that the device would seamlessly switch platforms depending on the RF conditions as the user drove. Each platform had a weighting. DAB was most important, followed by FM then IP. This is because DAB offers the best audio quality, FM provides long range radio reception and then the internet stream last because it consumes cellular data.

1.4.4. IRT DABerry

The basis of the prototypical implementation of a hybrid Radio receiver is the hybrid radio platform “DABerry” of the IRT.

The feature set of the DABerry prototype is:

- DAB/DAB+ reception through a USB DAB Stick
- Fulls support of DL/DL+, Slideshow and SPI.
- Bookmark a radio program in EPG (DABerry automatically switches channels)
- Recording of live signals (scheduled or manual)
- Time-shifted radio with segmentation based on DL+

Hardware and Software of DABerry

The Raspberry Pi is an all-in-one minicomputer platform, about the size of a credit card. Equipped with a Broadcom BCM2853 ARM System-On-A chip with 700 MHz and all necessary interfaces such as ethernet, USB and an analog audio connection, it can be put into operation directly without further components. The operating system used is a headless Debian Linux system installed on an SD card. The Elgato EyeTV USB receiver with Siano 2230 chipset is used for DAB radio reception.
Figure 12: DABerry hardware

To operate the hybrid receiver directly, a touch screen, which is connected via the GPIO connectors of the Raspberry Pi, is used. The touch screen is suitable for basic operation of the receiver with which the transmitter can be selected and the volume adjusted. In addition, the current slideshow image and Dynamic Label text are displayed on the screen.

However, the DABerry only unfolds its full range of functions and operation when used with another device on the same network and running an HTML5-capable browser. This can be a smartphone, a tablet or a PC. Simply calling up the DABerry IP address in the browser opens the DABerry HTML5 user interface which provides a comfortable use of the full functional range of the hybrid system.
In order to give the development platform the external appearance of a radio receiver, the entire system was built into a housing including audio amplifier and loudspeaker.
The DABerry platform achieves hybridity by making a RadioDNS RadioEPG/SPI request for each DAB radio service found by a station search. If the request is successful, the Extended Service Information (XSI) file for the service is retrieved. This XSI file provides extended information about the service, such as station logos, as well as information about the possible additional reception paths for the respective service. This makes it possible to identify additional reception paths, for example via IP or analogue FM reception. Since information about different reception paths is now available for a radio station, the receiver can select the qualitatively best and also cheapest reception path based on the local reception situation and the available connections.
The RadioEPG XSI file can also provide information about any services that can only be received over IP. These channels often offer a programme for a special target group. For example, private broadcaster Antenne Bayern currently has three radio stations with conventional radio distribution via DAB and, in addition, twelve other radio stations focusing on a specific genre with exclusive distribution via IP. This makes it possible to add further programs to the DABerry service list and further increase the program variety.

**Skipping/Time shift**

The DABerry uses the "ItemRunning" and "ItemToggle" bit contained in the Dynamic Label Plus (DL+) data. The ItemRunning bit is used to signal the beginning of a programme, e.g. the morning programme lasting several hours. This bit remains constant throughout the entire programme and changes at the beginning of the next programme. The ItemRunning also jumps over when the broadcast is interrupted. This is the case, for example, if a safety-relevant ghost driver message is sent.

The ItemToggle bit skips at the beginning of each part of a program. So this bit changes with each new song and also with each new contribution within a transmission. These two indicators are used to implement the "skip" function. If the listener presses the skip button during playback of the linear radio programme, either a title from the local music database or an on-demand title recommended by the recommendation engine is played, depending on the situation. The linear radio program is immediately stored locally (time shift) together with all meta-data such as slideshow and DLS. Using the ItemToggle bit, the paused linear program is now always marked and segmented in the background at the beginning of a new post. When the replacement content is over, the cached linear program continues to play from the first marker.

The listener has thus skipped only the last title and can then continue listening to the now time-shifted transmission. If the skip function is used again, the system first jumps to the next markers in the delayed linear program. Only after the last selection is the live program replaced with a personalized title.

The skip function ensures that the listener as a whole remains with the program and thus with the station by exchanging only individual, unwanted
content for personally preferred content and not switching to another station.

Figure 15: Buffer segmentation based on DL+

**Recording**

Based on timers or EPG data, the DABerry is capable to record radio programs (or parts thereof) to its local storage for later consumption by the end user.

**RadioWEB**

To further advance the hybrid radio idea, a new RadioDNS service called RadioWEB was specified and submitted for standardization as part of the DABerry developments.

RadioWEB defines how certain parts of a hybrid radio user interface can be replaced by an HTML5 component defined by the broadcaster. The specification also contains a programming interface (API) whereby certain parameters can be called up and also functionalities of the receiving device can be controlled.
A RadioWEB service is located the same way as any other via RadioDNS as a "radioweb" service record, which resolves to the address of a RadioWEB application information document in an XML format. This contains all the necessary information for executing and displaying the Web application.

Figure 16: RadioWEB service from AntenneBayern embedded in DABerry UI

1.5. METADATA PLATFORMS

Metadata platforms are commonly used for metadata management as well as for the management and analysis of other data. This other data is generally referred to as content data. Most often, these terms are used in relation to digital media, but it also encompasses other forms of content and content sources, which can be described with metadata tags, such as catalogs, dictionaries, or taxonomies.

But besides organizing content, the creation of metadata schemes and the development of systems in which this information can be managed, is also crucial for the success of discovering knowledge within. Indeed, data without context is of limited value, while metadata provides exactly the kind of context for different content items.

In this context, there are a lot of different research questions, which have been in focus of scientific research during the last years. In general, they mostly deal with different topologic and distributed search concepts, metadata organization and representation as well as with the process of
creating knowledge from this information, e.g., to create recommendations and to make tangible usage of aggregated data schemes.

As a result, there are several existing metadata management systems, with their advantages and disadvantages within different usage scenarios. In the following, we will highlight a range of different concepts, which provide important insights into the process of metadata collection, organization, aggregation as well as into different approaches of data mining and knowledge creation.

- Aldea et al. present the development of a so-called knowledge management platform for web-enabled environments featuring intelligence and insight capabilities [KOK03]. Therefore, the authors make use of Multi-Agent Systems as well as of ontologies. The automatic evolution of dynamic ontologies requires the action of a collection of agents to extract information and discover links using classification and learning technics. Moreover, the agents are designed in a multi-purpose style, which allows the goal to maintain periodical access onto the ontology and also to support search functions. Conceptually, similar documents or items get clustered into categories and information can than be retrieved by statistical approaches. Discovery of new knowledge leads to modifications within the ontology by pruning irrelevant sections, refining its granularity, and testing its consistency.

- A hybrid recommender system has been developed by Cantadoret et al. known as News@hand [CAN08]. It provides online news recommendation services using semantic technologies. There are domain ontology concepts in the system which are used to describe the news items as well as user preferences. The system supports two models, a content-based model and recommendation model based on collaborative filtering. The choice of model depends upon the similarity between the item description and user profiles as well as within semantics in the concepts involved. In the authors work, a model has been evaluated which will take care both of the users long-term interest and their current interest. To preserve a temporal state of interest of a user, a model has been developed which can personalize the order of visibility of news items. Thereby, the user’s current interest
becomes protected, for the model can reorder the news items which have been developed.

- In a related context, Vallet et al. propose a model for exploitation of an ontology-based Knowledge Base to improve searching over large document repositories as well as over large scale document repositories and to provide better search capabilities which yield in a qualitative improvement over keyword-based full-text search [Val05]. To manage large size information, a classic vector space model has been proposed, upon which a ranking algorithm has been defined. The result of performance tests of the proposed concept is directly proportional to the amount and quality of information present in the Knowledge Base. Though efforts have and are being made in the area of automating ontology population and text annotation, there still exists a limitation on the completeness of the available ontology and the Knowledge base. This situation indicates that the techniques of keyword based search and practice of recall will prevail, as long as ontology information is unavailable or incomplete.

- Alcantara et al. emphasize on how to retrieve thousands of web pages, how to filter them, to analyze the information and how to integrate them into a knowledge repository [ALC05]. To address these challenges, they used a system named as MASH (Multi Agent Search Engine). MASH is able to detect web pages related to domain ontology and calculate the relevance of each one. In MASH four different types of agents were identified: 1) one user Agent (UA), which interacts with the user, 2) internet agents (IA) which access, retrieve, rate and filter the information from the web, 3) weight agent (WA) that supports the search and supplies alternative queries when requested, and 4), coordinator agent (CA) that rules the overall process, particularly during splitting and amalgamation of the tasks performed by the IA. Finally, both the input and the output of the search process, are handled using a web interface, where the user can edit, create, retrieve, import and export ontology as Resource Description Format files.

- L’ecu’e emphasizes an approach that would serve the interest of the end users [LEC10]. In order to achieve this, a hybrid approach involving pure and classic collaborative-filtering methods and a semantic
content-based modelling has been presented. The former method involves automatic recommendation of services depending upon the users with similarity in their profiles, preferences and experience. The latter focuses on analysing the semantic similarity of services, following a Description Logic based reasoning.

- For content-based retrieval of cultural objects, the approach used by Govedarova et al. involves characterization of metadata and uses domain ontology [GOV08]. This concept of ontology is used in a case-based reasoning (CBR) process to simplify the complicated, multi-relational case structures. In order to create a standard system some cases were composed consisting of different attributes with simple data types such as string, integer etc. To evaluate the vitality of any attribute in representing the concept of ontology, a feature of concept data type supported by the jCOLIBRI framework has been used. In this architecture, the value of any attribute will correspond to the instances of the linked concept. Besides this, another feature of jCOLIBRI, which has been of advantage is its two layer organization of the case base. Here, the metadata for each entity to be described is created by observing the details of their physical representations in the Electronic Catalogue (Persistency media) and is stored in the case base (In-memory organization). The two layers of the case base are connected via connectors. These connectors read the value stored in the database and return them to the application by assigning a value to the attribute.

- The authors of [BIE13] propose two new families of inconsistency-tolerant semantics, that approximate the Consistent Query Answering (CQA) semantics respectively, and converge to the CQA semantics in the limit. They studied the complexity of conjunctive query answering under the proposed two parameterized inconsistency-tolerant semantics, and showed a general result for all first-order rewritable ontology languages.

All of these concepts provide important thoughts towards the development of a distributed metadata platform, capable of aggregating content from various metadata sources, including in the context of hybrid radio. Moreover, they allow the derivation and specification of holistic data models and of their needed representations for the following process of its structured and
automated analysis on basis of learning and knowledge discovery algorithms.

1.6. DISTRIBUTED DATABASE TECHNOLOGIES

A large number of distributed data storage technologies exists. We focused on those suitable for document storage and retrieval. In order to pre-filter and evaluate the specific technology landscape, we defined a set of requirements:

→ Service search equivalent to FM/DAB frequency scan
→ No centralized architecture (cost, no single point of failure.)
→ Basic search for services (keyword, genre)
→ As little bootstrap information as possible (What does a radio need to know for scan initiation?)

Based on these requirements, several technologies were selected and evaluated in detail.

1.6.1. Distributed Hash Tables (DHTs)

A distributed hash table (DHT) is a fully distributed hash table based on peer-to-peer (P2P) principles. It is usually used to store a specific resource accessible by its key in an efficient and decentralized way. The key can be derived from the resource’s content via a hashing algorithm (e.g. MD5). Data is not only stored on a specific node, but is distributed among a certain subset of nodes dependent on a certain routing strategy. Thus, if one node fails, data is still accessible.

Different protocols exist for various specific use cases:
DHTs have the following disadvantages:

→ **Bootstrap**: There must be at least one known node in order to enter the DHT network. This node must be fault tolerant and highly accessible.

→ **Complex Query Processing**: The easiest way to query is simply using the resource’s key. But if a query should search for a set of resources not by key but by a specific resource attribute, more sophisticated strategies must be implemented on top of the DHT overlay network. In summary, it is much harder to implement complex queries efficiently.

We’ve prototyped a Kademlia based approach for metadata dissemination. Based on its evaluation, we’ve dropped a DHT approach for a more centralized but hybrid approach based on a federation of servers.

### 1.6.2 Distributed NoSQL Document Stores

- **MongoDB** is a document focused NoSQL database, written in C++. Due to its document oriented working style, it is possible to manage collections of JSON-like documents. This allows many applications to model data in a more natural way, as the data can be nested in
complex hierarchies, while always being queryable and indexable. Due to its design as a document store, it lacks of modern approaches of distributed search within different server nodes within a fragmented consolidation of entities. Therefore, despite its document-oriented style of working, it is less suitable for a distributed search and metadata platform, as proposed within the HRADIO requirements.

- **CouchDB** Apache CouchDB is a database management system based on a document-centric database developed distributed under the terms of version 2.0 of the Apache license. CouchDB is mostly written in the programming language Erlang. The scripting language is JavaScript, which is interpreted by SpiderMonkey on the server side. The goal of CouchDB is to combine the simple data model of a document-oriented database with the scalability and performance of a professional relational database. For this, CouchDB must be usable on multi-core systems and in multi-server environments. One disadvantage of CouchDB is, similarly to MongoDB, its NoSQL-based design, which leads to requirement of trained personnel for its maintenance. Furthermore, the design is not relational, which means there is no possibility of writing or constructing automated queries in order to select specific data from several tables. Another big issue are high demands for memory and physical resources if scalable high performance services are needed. Another issue is the need for the manual updating process.

- **Apache Solr** is a highly reliable, scalable and fault tolerant enterprise search platform, which provides distributed indexing, replication and load-balanced querying, automated failover and recovery, as well as centralized configuration. Moreover, Solr has both, individuals and companies who contribute new features and bug fixes. Solr is completely Open Source and community driven, while its ecosystem lacks of tools and adoption of different programming languages, for it is purely Java based. In its current implementation, it is also not possible to implement individual entity nodes representing self-contained clusters.

- **Elastic Search** is a search engine based on Apache Lucene. The program written in Java stores the search results in a NoSQL format (JSON) and outputs them via a RESTful web interface. Elasticsearch is
the most popular search server besides Solr. It enables simple operation within a computer network for implementing high availability and load distribution services, which most fully meets the requirements of the HRADIO metadata platform.

In general, an Elastic Search ecosystem consists of a collection of nodes that hold data. It provides indexing and search capabilities across all nodes and is identified by a unique name. Each node is a single server that is part of the cluster. It stores the data and participates in the cluster’s indexing and search capabilities. The index itself is a collection of documents with similar characteristics, such as customer data or order data. It’s like a relational database, with mappings that define multiple types. Additionally, there can be multiple types within an index. For example, an application has the GetCount index, which has different sections like active user count and inactive user count. Due to its ancestry to Lucene, which is a full-featured information retrieval library, it provides a powerful full-text search capability. Moreover, it is also document-oriented and therefore capable, to store complex real-world entities as structured JSON documents. It indexes all fields by default, with a higher performance result compared to similar approaches. In addition, Elastic Search is scheme free and accepts JSON documents, and can detect data structures when indexing data, making it automatically searchable.
2. DERIVED TECHNICAL REQUIREMENTS AND GAP ANALYSIS

2.1. DERIVED TECHNICAL REQUIREMENTS

This section starts with an overview of the HRADIO derived system requirements. In the process of WP2, an extensive set of user scenarios has been developed, which are well suited to demonstrate the power of hybrid radio. A detailed description of the process and its results can be found in deliverables D2.1: “HRADIO User Scenarios” and D2.2: “User Evaluation Report”.

Based on the list of selected HRADIO user scenarios, deliverable D2.3 ”Initial User and System requirements” analysed these scenarios and initially derived a set of system requirements in order to have a consistent set of technologies which are necessary when implementing the HRADIO user scenarios.

The main task of work package 3 is the development of basic technologies that are later used in work packages 4 and 5 to implement the HRADIO use cases for the pilots. To structure these developments, work package 3 has been split into three different tasks:

**T3.2:** Development of a distributed communication and metadata platform. The aim of the work is to develop uniform platform and interfaces for the distribution of service and program metadata, to integrate recommendation systems and to collect usage data taking into account the privacy needs of the individual end users.

**T3.3:** Development of Android, iOS and HTML/JavaScript client libraries for HRADIO pilot development on common client platforms.

**T3.4:** Development of a cheap but fully featured DAB+ radio playout for pilot development and in–house test purposes.

In a first step, these requirements are further classified regarding the following aspects:
To which task in WP3 is the requirement can be assigned.

Is the requirement radio specific or in general an implemented feature on mobile devices?

Especially the classification whether a requirement is radio centric or not plays an important role for the definition and the scope of the tasks in WP3.

Radio specific requirements will be in the direct scope of the technical development work in WP3 either as a new implementation or (more likely) as an integration of existing work into a uniform set of libraries for the later use in application development.

General system requirements are not the direct focus of the WP3 developments. However, the requirements analysis has shown that these requirements are critical for the respective use cases and needs to be present on the target platforms to enable implementation. They have no direct relation to the radio world and are usually used for other purposes on the terminal. Often these technologies are already available on the end devices and are available to application developers through public APIs. If necessary, an adaptation or integration into the WP3 libraries is necessary.

The following table shows the derived system requirements from D2.3 together with the introduced assignment to WP3 tasks.

<table>
<thead>
<tr>
<th>Req</th>
<th>Description</th>
<th>WP3 Task</th>
<th>Radio/General</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAB-DL+</td>
<td>The DynamicLabel and Dynamic label+ (DLS/DL+) functionality is used for text messaging integrated with radio services. Messages usually contain title and artist information, but can also include weather, traffic and general information related to the radio service (e.g. web links, hotline information ...). In addition to text information and tagging information, DL+ specifies flags that can be used to signal accurate trigger events, when program elements are started or temporarily interrupted (e.g. due to announcements).</td>
<td>T3.3/T3.4</td>
<td>Radio</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.etsi.org/deliver/etsi_ts/102900_102999/102980/01.01.01_60/ts_102980v010101p.pdf">http://www.etsi.org/deliver/etsi_ts/102900_102999/102980/01.01.01_60/ts_102980v010101p.pdf</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DAB–SPI | DAB Service and Programme Information (SPI) provides detailed service and program information to radio receivers. The WorldDAB EPG standard, with additions from the RadioDNS EPG spec, specifies a XML document structure which allows the provision (either binary encoded in Broadcast or over IP as XML file) of structured data regarding service, schedule and on-demand content data. 

http://www.etsi.org/deliver/etsi_ts/102800_102899/102818/03.01.01_60/ts_102818v030101p.pdf

Normally SPI information is used to describe program data for the upcoming days in the radio schedule. Additionally, DAB–SPI supports various bearer elements in the schedule, allowing the signalisation of on-demand content in addition to the time-based schedule.

Additionally the SPI specification also allows the signalisation of geo location information for service and program data. This enables location-based personalization of recommendations.

| T3.3/T3.4 | Radio |
| DAB-SLS | The slideshow service provides program-related image data (JPEG or PNG) either over the DAB broadcast system or via IP (STOMP or HTTP). Content providers can attach URLs to the transferred images which can provide alternative locations/versions of the image data or more related information. This feature is known as a categorised slideshow. Alternative URLs can be requested by the device including the current geolocation in the HTTP header information. |

http://www.etsi.org/deliver/etsi_ts/101400_101499/101499/03.01.01_60/ts_101499v030101p.pdf |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RadioTAG</td>
<td>RadioTAG provides a simplified interaction method for radio, particularly in challenging environments such as when driving a vehicle. Here a user can perform a single physical action to express interest in the content they are hearing on the radio and schedule their exploration of that content for a later point in time.</td>
</tr>
</tbody>
</table>

http://radiodns.org/wp-content/uploads/2014/02/rtag01_v100_draft_7.pdf |
RadioWEB specifies a HbbTV-like service for mobile radio devices with browser technology. Based on the general FQDN lookup mechanism of the RadioDNS system, RadioWEB defines a special DNS record for signalling the entry point to an HTML5-based application accompanying the radio service. Just like HbbTV, RadioWEB defines the application signalling, an application life cycle and a JavaScript-based API for access to radio hardware resources such as tuners.

In HRadio, the API part of RadioWEB will be covered and extended by a dedicated JavaScript API, while the signalling and lifecycle elements remain relevant. The draft specification proposal can be found here:

https://drive.google.com/file/d/16crbYzNjm6mIDSWYIoHliyp6jww79Pwr/view

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RadioWEB</td>
<td>RadioWEB specifies a HbbTV-like service for mobile radio devices with browser technology. Based on the general FQDN lookup mechanism of the RadioDNS system, RadioWEB defines a special DNS record for signalling the entry point to an HTML5-based application accompanying the radio service. Just like HbbTV, RadioWEB defines the application signalling, an application life cycle and a JavaScript-based API for access to radio hardware resources such as tuners.</td>
<td>T3.3/T3.4</td>
</tr>
</tbody>
</table>
| Device Pairing | This functionality enables the preservation of context and state over different devices or usage scenarios, and requires a mechanism to associate different entities to a common token/user account. Possible candidates:  

EBU CPA: [http://www.etsi.org/deliver/etsi_ts/103400_103499/103407/01.01.01_60/ts_103407v010101p.pdf](http://www.etsi.org/deliver/etsi_ts/103400_103499/103407/01.01.01_60/ts_103407v010101p.pdf)  

Shared State from EU MediaScape project: [http://mediascapeproject.eu/files/D4.3.pdf](http://mediascapeproject.eu/files/D4.3.pdf) | T3.3 | General |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Time Shift player</td>
<td>This functionality provides a local timeshift buffer on the device, and is accessible either via usual time shift commands (Play, Pause, Rewind, Forward) or via a DAB-DL+ segmented buffer for content-based Skip forward or Skip backward commands.</td>
<td>T3.3</td>
<td>Radio</td>
</tr>
</tbody>
</table>
| Server-based timeshift component | This functionality enables a timeshift buffer on a remote server in order to be shared between different paired devices. The server might use an anonymous shared token (e.g. via NFC or QR code) to achieve this requirement. However, a service provider might require a user login for such a premium service offer.  

The same UI functional requirements as for Local Time shift will be supported. | T3.3/T3.4 | Radio |
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Section</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recording</td>
<td>This functionality enables scheduled or time-based recording of radio programmes. The Recording function needs to manage finished and scheduled recordings. Some radio services may require restrictions on recording.</td>
<td>T3.3</td>
<td>Radio</td>
</tr>
<tr>
<td>Geo location access</td>
<td>This functionality enables access to the current geographical position of the device.</td>
<td>T3.3</td>
<td>General</td>
</tr>
<tr>
<td>User Account Management</td>
<td>This functionality enables user account management for server-based time shift and recommendation scenarios.</td>
<td>T3.3</td>
<td>General</td>
</tr>
<tr>
<td>Recommendation-engine</td>
<td>To deliver a personalised linear or on-demand radio service, a recommendation engine selects alternative content based on a personal user profile. The recommendation functionality can be either local to the receiving device or server-based. Multiple usage scenarios for radio enable users to listen to radio in a variety of situation. Recommendations based only on matching content are often inadequate, thus a combination of content and context recognition is necessary.</td>
<td>T3.2</td>
<td>Radio</td>
</tr>
<tr>
<td>Environment Sensors</td>
<td>This functionality enables detection of various local environment situations (e.g. person in room).</td>
<td>T3.3</td>
<td>General</td>
</tr>
<tr>
<td>Traffic information</td>
<td>This functionality enables general access to traffic information such as</td>
<td>T3.3</td>
<td>Radio</td>
</tr>
</tbody>
</table>
### 2.2. MAPPING OF REQUIREMENTS TO ACTUAL STATE OF THE ART TECHNOLOGY

#### 2.2.1. T3.2 requirements

In the current set of system requirements, T3.2 shows only two assignments (“Recommendation engine” and “Standard Radio functionality”). However, this means that T3.2 must support the build-up of a distributed, robust and scalable metadata search engine for service and content search, utilizing a...
comprehensive radio centric data model and a storage system as well as import facilities to attach the HRADIO infrastructure to existing metadata sources. The state-of-the-art analysis showed that a wide range of technologies is available to realize an implementation of this feature set. In a first approach, ElasticSearch, as a storage and search engine which supports a distributed setup, seems to be the most reasonable choice. However, in the second iteration of the HRADIO user and system requirements it is expected (also with the feedback from the first pilot phase), that more detailed requirements especially with the regard of privacy and recommendation technologies will emerge.

2.2.2. T3.3 requirements

This set of requirements needs to be reflected in the implementations of the HRADIO client libraries.

Text messages (DAB-DL+), images (DAB-SLS) and program data (DAB-EPG) have long been part of the feature set of radio services via DAB. The transmission of text messages in analogue FM is possible but is often not implemented in the end user devices, although, for example, the German public broadcasters broadcast text messages as a regular service. DAB-SLS as well as DAB-SPI have no broadcast counterpart on analogue FM systems and can only be supported via the hybrid RadioDNS signalisation in analogue transmissions. The usage of these technologies for the implementation of HRADIO user scenarios should be straightforward, as implementations, tools and know-how exist.

IRT, with its DABerry prototypes as well as the Radioplayer car plattform and the existing LG DAB enabled Smartphone, support this technologies well and implementations exist and can be reused.

RadioTAG as well as RadioWEB are available as draft specifications only, but as the state-of-the-art section in this deliverable reveals, prototype implementations exist. In the case of RadioTAG, a publically available open source implementation is provided as a joined project from EUB and BBC⁵. For RadioWEB, IRT implemented a fully working prototype as part of the DABerry developments and the technical feasibility has been demonstrated.

⁵ https://github.com/ebu/radiotag.js
Time-shift (local) also has been demonstrated in IRT DABerry prototype as well as in many other even commercially available products. In combination with DAB-DL+ functionalities, a content based segmentation and accessibility of the local time shift buffer is realizable and has been demonstrated. Server based time shift needs additional components in the backend to provide the proposed decentralized and device independent usage, however for the work in T3.3 the API implementation for the local and server-based time shift component will look quite similar.

Recording is a well-known problem and implementations exist.

Traffic Information is provided by established specifications for analogue (TMC) as well as in digital broadcasting (TPEG). Different (Java/C++) implementations exist (e.g. IRT TPEG decoder) and can be reused.

Multituner systems are common in the radio industry especially in car environments.

Standard Radio Functionality has been demonstrated in the IRT DABerry prototypes, in Konsole clients and in Radioplayers Car prototype.

General device requirements:

Device Pairing is a general problem for many mobile application developers. Pairing of end user devices either device to device, or device to web service is often required and common solutions exist. In media centric platforms such as HbbTV, IRT for instance has developed a Second Screen Framework which works with standard web technologies and uses QR codes for pairing. In the current HbbTV 2.0 Specification, second screen scenarios are included and make use of DIAL, WebSockets and related specifications. The EBU CPA covers the same use-cases specification in an international ETSI specification.

Geolocation access in mobile/stationary devices using the GPS technology is a common task for developers and well understood.

User account management is also a general problem of end-user applications on mobile and stationary devices. Many open and proprietary solutions exist and are available for HRADIO applications.
Environment Sensors. One of the scenarios (see D2.3 “S13 Automatically Adapted Radio”) requires the availability of more sophisticated sensor technology in the device (e.g. recognize and locate person in the room or background noise detection) which is beyond the scope of HRADIO. However, it is expected that in the future, real time object or person detection becomes a common functionality of the device platforms targeted by HRADIO.

Voice control is becoming a popular mode of user interaction in media applications in general. Devices such as Amazon Alexa or Google Home demonstrate a nearly natural speech communication between a person and a device. The development of voice recognition and control technology is not in scope of HRADIO, however its application definitely is.

2.2.3. T3.4 requirements

DAB-DL+, DAB-SPI and DAB-SLS are requirements which are general end-device technologies. However, for the envisaged HRADIO lab playout system developed in T3.4, the generation of standard conforming dynamic label, slideshow and service and programme information data is absolutely necessary. Tools and software libraries for the generation of the data services exists (e.g. from Opendigitalradio.org) and could easily be used for the purposes of the HRADIO project.

RadioTAG and RadioWEB are purely based on the HTTP/HTTPS protocol family and can be handled with a standard HTTP server.

Server based Timeshift, for a pure radio centric server based time shift component (esp. for DAB+ based radio content) currently does not have known supporting technology. For TV, such technology has been developed and used. It has been demonstrated in the DABerry prototypes from IRT that on device time shifting of DAB+ based radio signal is technically feasible. In the related T3.4 work, these developments need to be ported from the Linux/ARM based RaspberryPI platform to a more common Linux/Intel based server technology.
3. FINAL HRADIO SYSTEM ARCHITECTURE

Radio is often reduced to the simple equation of audio services. Sure enough, the audio signal is by far the most important component that the broadcaster delivers. However, surrounding the audio services is a rich set of additional metadata and functionalities such as slideshows, text messages, EPG data, alarm announcements and traffic information, service following signalizations and many more.

Listeners want to listen to their favorite radio station regardless of whether it is received via FM, DAB+ or IP. At home in the WLAN, IP is usually no problem. On the road in the car, DAB+ will be a better choice. However, when listeners drive out of the regional coverage area, IP may become the best option again. Similarly, in the car, additional functions such as service following or alarm announcements are extremely important.

On top of network continuity, users now more and more expect radio to converge with streaming services such as Spotify, with features such as integrated pictures, and pause and skip functionalities.

In WP3, the technical foundations of the HRADIO application development have been developed with a focus on three areas:

- The **Playout** side: To provide playout tools enabling the quick and easy provision of radio signals and time independent streams to HRADIO client applications.

- The **Client APIs**: To provide a radio centric set of application functionalities, which can be used by application developers.

- The **HRADIO Communication Platform**: To provide access to service and program data as well as service recommendations and privacy preserving collection of usage data.

The figure below shows an overview of the three different tasks and their components.
Core of the developments in WP3 was the client development. In T3.3, a set of radio service centric client libraries for the most significant platforms have been developed. The work of the WorldDAB OMRI taskforce has served as a basis for these libraries. After the experiences with the first DAB equipped smartphone (LG Stylus), IDAG and WorldDAB started standardisation work on a platform and technology agnostic API for mobile devices. Goal of T3.3 was primarily to implement a fully working implementation of this OMRI API for the most common mobile platforms. A secondary goal was to provide feedback to WorldDAB and W3C on the standardisation of these APIs based on the experience gained during the implementation phase.

While pure broadcast radio can operate on completely unconnected devices (all necessary service information, data services and signalisations are part of the broadcast signal), pure IP or hybrid solutions require the provisioning of that data over an IP connection. On top of that, the user scenarios in HRADIO focus on the delivery of a combination of broadcast radio together with individually delivered personalized content. To achieve this, a distributed communication and metadata platform was developed in T3.2. Core functionalities of that platform include the provision of metadata for service and content-related search of radio services, functionalities for individual recommendations and a privacy preserving collection of usage data.
As a third task, T3.4 a lab playout system to provide the HRADIO project partners with a common platform for development and testing of HRADIO user applications has been developed. This task started from the work of the OpenDigitalRadio.org initiative, which provided a set of open source tools for multiplexing and modulation of digital radio signals based on software-defined radio (SDR) technology.

Furthermore, for devices which cannot receive broadcast signals directly, or for which no external receiver hardware is available, in T3.4 a DAB to IP splitter, has been developed, allowing the streaming of a single service DAB stream over IP to such devices.

3.1. CLIENT LIBRARIES T3.3

These requirement - from a "simple linear" service to a mix of different services and uses cases make the development of radio applications extremely complex. In order to reduce this complexity for the application developers, this task focused on the development of a set of client libraries, which enable application developers in the HRADIO project to shift focus away from technology towards the development of novel hybrid radio scenarios. In order not to face the complexity of integrating too many different technologies (DAB/DAB+, Shoutcast, HLS, WebSockets, ...) an important design decision is the use of DAB+. DAB+ is standardised in almost all facets and additional data services. Slideshow (including linking), DynamicLabel+, ServiceFollowing, Alarm Announcements are defined in the DAB+ multiplex and are coded and transmitted synchronously and optimally "in band". If an application developer knows DAB+ and has the possibility to use DAB+ on all platforms (Car, iOS, Android and Web), he does not have to constantly rethink in the technical context and can concentrate on the functionality. To realize this, the following architecture has been developed in the HRADIO project

The basis for the developments is the OMRI specification. OMRI (Open Mobile Radio Interface) is an API definition that enables application developers to use radio services, regardless of the underlying technical platform (DAB+, FM, IP, HD Radio...). OMRI is standardized by WorldDAB+ (ETSI TS-103632 V1.1.1) and the HRADIO project aims to implement OMRI for the major platform Android as well as for iOS and the Web.
The project mainly works on the Android platform, since Android has the largest user base on mobile devices and, due to developments such as Android Car and Android AutomotiveOS, is also gaining more and more attention in the automotive sector. The OMRI implementations in HRADIO are based on a stable ANSI C++ 11 framework for demultiplexing and decoding DAB+ Ensembles.

![OMRI Application (Android, iOS and JavaScript)](image1)

![OMRI API](image2)

![JNI Layer for Android](image3)

![Swift/C++ Layer for iOS](image4)

![WEBAssembly/ emscripten with JS/MSE](image5)

![Native C++ IRT DAB Libraries](image6)

![DAB-Multiplex (DAB/IP)](image7)

![FIC-Packets](image8)

![MSC-Packets](image9)

**Figure 18: Client libraries architecture**

### 3.1.1. Android

For the Android work in T3.3, as a first step, the native C++ implementation will be ported to the ARM-based Android platform. In a second step, a JNI (Java Native Interface) layer will be implemented to provide access the native C++ functionality from the Java-based OMRI API. Together with this document, deliverable D3.3 describes this work and a working implementation on Android has been released as a beta version.
3.1.2. iOS

As iOS development allows the direct integration of C++ based developments into application projects, contrary to Android, no additional interface layer is needed and an Objective C- or Swift-based version of the OMRI API can easily be implemented. However, tuner devices for DAB signals are very unlikely to be found in iOS devices. Therefore, the iOS version of the HRADIO client libraries will focus on IP-based reception of digital radio signals as outlined in the T3.4 description.

3.1.3. HTML/Browser.

In order to use HRADIO clients implemented as web applications, HRADIO will also develop a pure HTML/JS version. DAB-data is either received via a USB receiver or streamed via IP. Decoding of DAB-data has already been implemented in C++. To avoid re-implementing this DAB-decoder in Javascript, the current plan is to investigate to transpile the existing C++ code into Javascript. This makes it possible to use the same code base for all three platforms Android, iOS and client-side web applications.

Web assembly (WASM) emscripten is a relatively new technique to enable web development with other languages beyond JavaScript. In general, there is nothing wrong with JavaScript, but for many problems it is not necessarily the first choice. In addition, JavaScript transfers the source code as part of the page to the browser, where it must be parsed before execution. Here web assembly promises a small and slim bytecode that can be loaded and executed quickly. First supported languages are C/C++. In the meantime, however, Rust, C# or Go can also be used. The browsers of Mozilla, Google, and Apple support this technique and with the compiler tool “emscripten” it is relatively easy to bring an existing C/C++ code base into the browser. In our HRADIO web implementation we use the same C++ code base as for Android and iOS translated with emscripten in the browser. Once the ensemble data has been loaded and decoded, the web developer can also use Slideshow, DynamicLabel and others as usual in his JavaScript world. The audio data can be played back in the browser through MSE (Media Source Extensions). Currently, there are still incompatibilities regarding the audio format in DAB+.

As a workaround, in addition to the DAB+ libraries, an AAC decoder, which
supports the DAB+ AAC stream (960 frames/AU), also needs to be run in the browser via emscripten.

![HRadio system architecture for web applications](image)

**3.2. LAB PLAYOUT T3.4**

The decision to use DAB+ as the technical basis for HRADIO application development means that a DAB+ signal must of course be available in all usage scenarios. In general, the OMRI implementation supports the usual DAB+ USB receiver sticks. This means that DAB+ reception is always available. On mobile devices, however, the external receivers represent a foreign body and are quite unwieldy for the user, especially with the antenna required for reception. But in places where no DAB+ signal is available or reasonably to use, it should be possible to switch to IP reception – which fits the hybrid vision of the project. The HRADIO project therefore decided to use DAB+ and
transmit the multiplex in IP streams when radio services have to be transmitted via IP networks. In the DAB+ family of standards there is the so-called EDI standard for this purpose. EDI was originally designed to feed DAB+ ensemble data from the multiplexer to the broadcast sites. An EDI ensemble contains several individual radio services and usually has a bandwidth of about 2.5 Mbit/s. For the unicast transport of a single radio service to the client it is of course not necessary to transmit the whole ensemble. Therefore, the project implemented a server software that receives the complete EDI stream from the DAB+ multiplexers in the broadcasting houses and (according to the number of services included) provides individual single-service EDI streams for IP transport.

This technical solution has great advantages for all parties involved. The application developers always work in the same DAB+ world. The services that are used (next to the audio signal) are “in band”, synchronous and their formats and behaviour are standardised. A transmitter that already generates DAB+ services for terrestrial transmission can send them into the EDI splitter without additional effort and use them as an IP stream.

Given that an extensive support of data services is essential for HRADIO application functions, both platforms (DAB+ and IP) equally benefit from the efforts made in playout. The EDI splitter is implemented in Google GO as a very lean server service. Even on a low-end platform such as a Raspberry Pi, the reception, splitting and redistribution of a complete ensemble is no problem. As time shift of radio services is a central use case in the HRADIO applications, the timeshift function is either implemented as a local buffer on Android devices, which makes it possible to use the entire timeshift buffer as soon as the service starts, or moved to the server to reduce the load on clients.
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 was 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 is more complex.

In order to enable the different developments partners to reproduce test radio signals within a real-life scenario, a simple lab playout platform has been developed in T3.4.

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 a demonstration service with the full support of DAB data services. This has been demonstrated with the live generation of the test signals for the RBB pilot.

The following figure shows the architecture and the components of the lab playout system.
The Lab Playout system for HRADIO consists of four parts:

1. **MPD**: An open source audio playback software, for the simple playback of m3u playlists. In the previous installment of the HRADIO Lab playout a full featured radio software for scheduling and station management was used (LibreTime). However for the reduced requirements of the HRADIO lab playout during the pilots, a simple audio player such as MPD proved to be a better fit. For the HRADIO pilots which used real life services from VRT, RBB and BR, a full featured radio production platform wasn’t necessary and for the simple playlist-based approach during RBB pilot 3 the LibreTime platform was not flexible enough to be tailored according to the requirements of this pilot. Together with MPG the YMPD –HTML UI was used to provide a reasonably easy but powerful GUI for the operation. The audio playback is then controlled by the MPD, which outputs a HTTP stream with audio data as input for the DAB+ audio encoder.

---

[^6]: https://www.musicpd.org
2. **Opendigitalradio (ODR) 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.

3. **PAD Inserter**: For each playback event from the MPD player, this tool will get the file currently played back from MDP and can retrieve the necessary metadata (Song, Artist, Cover Art, ...) from the audio files (ID3 Tags), convert it into the appropriate format and send it to the Opendigitalradio tools. The PAD encoder will use this data to create the slideshow and dynamic label for DAB.

4. **ODR-Control**: 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 ODR Tools and Docker. 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 is also 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.

The playout system is published using a docker-compose file. Using docker and docker-compose, it enables an out-of-the box deployment of the whole playout using just one command, making it to distribute and nearly hardware independent.

**3.3. HRADIO CONTENT AND METADATA PLATFORM**

In addition to the libraries for client development, a further focus of the HRADIO project has been the development of a decentralized platform

---

7 [https://github.com/Opendigitalradio/](https://github.com/Opendigitalradio/)
8 [https://www.ettus.com/](https://www.ettus.com/)
9 [https://github.com/hradio](https://github.com/hradio)
based on open standards for the provision of service and program data, the collection of usage data (feedback channel) while taking into account the privacy of the users, and the custom-fit, content-based recommendation of radios. While it is easy to scan the available frequency bands in a classic broadcasting network (frequency scan), the Internet-based search for IP services requires the provision of a so-called aggregator that collects service information and makes it searchable. The HRADIO project uses already existing RadioDNS service information as a basis for a distributed search platform, which was designed for the discovery of IP services by HRADIO clients. The platform is divided into several modules: The importer module collects RadioDNS SI and SPI files, ICY tags embedded in IP streams and DynamicLabel+ data and enriches them with additional data from other sources (e.g. additional information about title and artist, etc.).

Another important aspect of hybrid radio technology is the collection of listener feedback, which can then be analyzed to improve the radio experience. Within the HRADIO project, a platform module was developed that collects usage data while respecting the privacy of the individual user. The underlying mechanism is based on the randomized response technique known from psychology, also used in Google’s Chrome Browser to collect telemetry data in a highly extended form\(^\text{10}\). This method offers strong privacy guarantees and still allows a meaningful evaluation of collected data.

Besides searching and finding radio stations, a powerful recommendation service is also part of the developed platform. To further protect the privacy of the user, recommendations are based on extracted station characteristics. Such characteristics include, for example, genre distributions derived from existing metadata, keywords extracted by Speech-To-Text algorithms and Natural Language Processing (NLP) methods, and reductions of the audio signal to compressed fingerprints using machine learning techniques.

The design as implemented contains an easy-to-deploy metadata storage and search node, a service component for the collection of usage data and a server component for service recommendations and consist of the components depicted in the following diagram:

\(^\text{10}\) https://www.chromium.org/developers/design-documents/rappor
Core Components:

**Metadata Search and Storage**: The search and storage engine processes document search queries and stores the documents. We currently use Elastic Search for this purpose. The design is implemented in a way that allows for other search & storage engines as well. Attached is a module to forward search requests to multiple search nodes (search federation). This architecture extension enables one of the central goals of the platform: distributed metadata search and storage.

**Music Metadata Storage**: Service to retrieve detailed metadata for specific items (mainly songs).

**Metadata Importer**: The flexible architecture allows for different data integration components. Currently, an importer for RadioDNS data and for DynamicLabel+ data (coming from a DAB Tuner, sent via a publish/subscribe broker using the STOMP protocol and converted to JSON) are designed and implemented. Both use the message bus for data transport to the metadata processor.
Recommendations: Content based recommendation engine for delivering radio service recommendations to the end user.

Privacy Preserving User Data Collection: Uses Privacy-Preserving Ordinal Response (RAPPORT) which guarantees that the user’s privacy is preserved by client-side data-distortion that is conducted before user data is transmitted to the server.

Dashboard: Provides the components for the creation and composition of data visualizations

Passwordmanager: Controls user access to safety-relevant operations

All of the above-mentioned components communicate over a **message bus** which connects all system components and allows for easy extensibility and scalability. We use RabbitMQ, based on the publish/subscribe communication pattern, as a message broker.

The data model used for metadata storage is derived from the RadioDNS standard and depicted in the following diagram:
Figure 23: HRADIO data model

Note that the elements in grey (ServiceGroup, ProgramGroup) are not yet part of the implementation since the HRADIO user scenarios do not require them.
Section 3 focused on the HRADIO approach of the basic three different building blocks: the Playout, Client and the shared Metadata Platform. This section outlines the combined usage of these components during the HRADIO Pilots and sample apps.

Figure 24: Overview of HRADIO components used for pilot applications

Services are provided into the system by:

- Real-life on-air services from the project partners and friendly broadcasters (e.g. Bayerischer Rundfunk and AntenneBayern). These services can either be received via the traditional DAB broadcast or via the HRADIO DAB-over-IP toolchain. The DAB-over-IP streams are also cached with the HRADIO SBT (Server Based Timeshift) infrastructure.

- For tests and dedicated pilot usecases (RBB Pilot 3) the HRADIO Lab Playout is used.

- Any other IP (Shoutcast) based stream are signaled through the HRADIO Metadata Platform and can be played in the HRADIO platform.
app, although without sophisticated HRADIO features such as segmented time shift or server-based time shift.

Service playback is handled by the HRADIO Client library implementations. Basis for that is the OMRI implementation. On top of the OMRI framework, helper and convenience libraries for RadioDNS parsing, RadioWEB integration, and time shifted radio playback (either server-based or local) are implemented.

Additionally, a set of application level libraries for Service recommendations, Item skipping (implemented for the Spotify service) and the Service and metadata lookup as well as the Usage Data retrieval, are available to application developers.

It is important to mention that the whole set of HRADIO libraries and infrastructure does not form a monolithic system which can only be used as a single building block. The HRADIO set of developments forms a toolbox, which provides a great amount of flexibility to application developer who can choose which parts to use for their specific scenario.

This is reflected in the choice and realization of the three pilot apps. The HRADIO platform pilot app acts as a general radio listening application which provides the end user with a universal tool to listen radio. It makes full use of the HRADIO Service and Metadata platform and integrates item replacement via Spotify in a horizontal way. The RBB and VRT pilot applications are bound to a broadcaster or to a specific service. Service search becomes irrelevant and item replacement is more focused on the non-linear offering from the brand. The set of HRADIO libraries and technologies fully support these different approaches.
5. CONCLUSION

In section 1, this deliverable analysed the current state of the art of HRADIO related technologies. The selection of technologies to investigate was based on their relevance for the HRADIO project and an overview of practical implementations in the field of hybrid radio receivers has been given. After that, in section 2 the derived system requirements have been further classified and mapped to the actual state of the art. Finally, in section 3, a full HRADIO system architecture for the three tasks in WP3 was given.

The state of the art analysis and the mapping of the system requirements to the related state of the art technologies showed that, for most of the system requirements, technological solutions exist and are available. In particular, the OMRI API supports a rich set of radio centric requirements and, as described in D3.3, is readily available to developers on Android, iOS and the Web platform.

Based on this OMRI implementation, which the project delivered, a comprehensive set of support and convenience libraries has been developed to help application developers to a maximum possible extend in their development tasks.

The high rated time shift use cases are enabled by local and server based timeshift components which both use the traditional DAB multiplex as basis for broadcast and IP streaming services. This technical solution enables the broadcasters and service providers to reuse DAB knowledge and existing DAB equipment and rely on the robustness of the system.

The HRADIO metadata platform provides a system for service and program metadata lookup, service recommendations and a sophisticated model from the privacy preserving usage data collection.

The lab playout enables the easy setup and demonstration of the HRADIO system.
REFERENCES


