EBU

OPERATING EUROVISION AND EURORADIO

# TR 042

# EXAMPLE OF AN END-TO-END OBA BROADCAST ARCHITECTURE AND WORKFLOW

SOURCE: ORPHEUS PROJECT

Geneva
May 2018

This page and others in the document are intentionally left blank to maintain pagination for two sided printing

## Foreword

Currently, only a few broadcasters are well advanced in their adoption of next generation audio (NGA) and are equipped with a full set of tools and applications that span from content production to distribution. Some successful pilot productions exist, but in general the broadcasting community has just started embracing NGA technologies. They need to experiment and learn.

In the past 30 months, ten major European institutions consisting of three EBU members, manufacturers and research institutions have developed and evaluated a complete end-to-end object-based audio broadcast chain centred on open standards and with a strong focus on real-world applicability. This is the EC ORPHEUS Project.

The ORPHEUS Project has published a Deliverable (D2.4) that describes its end-to-end reference architecture; an abridged version of this Deliverable has been made available to the EBU to create this Technical Report containing a concrete and practical example of an object-based audio architecture and workflow from production to broadcast & broadband delivery.

In Annex A there is supplementary information, obtained from several sources, which can aid the reader in his/her general understanding of the subject.


Grant Agreement No.: 687645 Research and Innovation action Call Topic: H2020 ICT-19-2015

The ORPHEUS project, funded by the EC within the scope the H2020 programme, is a research and innovation collaboration from European public service broadcasters, manufacturers and R&D institutions.

One of the major objectives of ORPHEUS was the specification of a reference architecture for end-to-end object-based production and broadcast workflows. To achieve this, the ORPHEUS consortium first specified a pilot implementation architecture which was the basis for the pilots in the project. Using an iterative process, and based upon the findings and lessons learned from these ORPHEUS pilots, the reference architecture was developed.

Unlike the pilot implementation architecture, the reference architecture is more format- and interface-agnostic, allowing it to be used more easily as a general guideline for other broadcasters.

Each stage of the reference architecture is described in detail, with identified components, interfaces, and protocols for a complete object-based audio workflow.

The EBU considers that this Technical Report can be beneficial to all broadcasters that need to

understand the technicalities and impact of putting in place a complete NGA workflow based on open standards. The EBU has recently published EBU Tech 3388 (EAR – EBU ADM Renderer) together with its open source implementation that it hopes will play a major part in promoting NGA integration and usage in workflows amongst broadcasters. A future EBU document or documents will specify ADM Profiles that target different aspects of the workflow (e.g. a production profile or a live profile) – effectively constrained subsets of the entire ADM – that will simplify its implementation and use in particular applications and parts of the broadcast chain. The "ADM Broadcast Emission Profile" described in the current Technical Report has arisen in the context of the ORPHEUS project by Fraunhofer IIS and is a good fit for MPEG-H and potentially other NGA technologies and workflows. The EBU is currently examining this profile and, if necessary, may adapt it to ensure that it exactly satisfies the needs of its Members.

# Contents



# Table of Figures

# Example of an End-to-End OBA[1] Broadcast Architecture and Workflow

| *EBU Committee* | *First Issued* | *Revised* | *Re-issued* |
|---|---|---|---|
| TC | 2018 | | |

**Keywords:** OBA, Object Based Audio, NGA, Next Generation Audio, ORPHEUS Project.

## 1. Introduction

This document forms an example of an end-to-end object-based audio architecture, defined by the ORPHEUS project consortium[2]. It contains the final reference architecture specification of ORPHEUS, which is the result of intensive discussions and several iterations over the duration of the project.

The architecture described here has been shaped by taking into account typical channel-based broadcast workflows but, more importantly, by also including the knowledge and lessons learned during the pilot phases and stages.

### *1.1 Motivation*

The object-based approach to media gives the most fundamental opportunity to re-imagine the creation, management and enjoyment of media since the invention of audio recording and broadcasting. One may argue that the core of the object-based approach to media is not new in itself, and that object-based audio has not been fully adopted so far, despite the fact that past and recent developments ([1] [3] [4]) have utilised some variation or parts of the object-based concept.

This is due to the fact that previous attempts did not cover the full creative and technical process including planning, editing, production and leading to the consumption of object-based audio. This has led to isolated and partial solutions, which did not take into account enough of the whole system required to unlock the full potential of an object-based approach.

The ORPHEUS project therefore opted for an integrated method, incorporating the end-to-end chain from production, storage, and play-out to distribution and reception. Only through this approach can it be ensured that the concepts developed are appropriate for real-world, day-to-day applications and are scalable from prototype implementations to large productions. In order to achieve this, the ORPHEUS project structure has been designed with a full media production chain in mind.

[1] Object Based Audio – see Annex A for an explanation

[2] https://orpheus-audio.eu/

## 2. Abbreviations

The following abbreviations are used throughout this report.

| | |
|---|---|
| AAC | Advanced Audio Coding |
| AAX | Avid Audio eXtension |
| ADM | Audio Definition Model |
| AES67 | Audio Engineering Society standard for audio over IP |
| API | Application Programming Interface |
| ATSC | Advanced Television Systems Committee |
| BW64 | Broadcast Wave 64 Bit |
| CDN | Content Distribution Network |
| CSS | Cascading Style Sheets |
| DAB+ | Digital Audio Broadcasting + |
| DASH | Dynamic Adaptive Streaming over HTTP |
| DASH-IF | DASH Industry Forum |
| DAW | Digital Audio Workstation |
| DRC | Dynamic Range Control |
| DVB-S | Digital Video Broadcasting Satellite |
| EBU | European Broadcasting Union |
| FIR | Finite Impulse Response Filter |
| GPS | Global Positioning System |
| gRPC | Google remote procedure call |
| HOA | Higher Order Ambisonics |
| HTML | Hyper Text Markup Language |
| IIR | Infinite Impulse Response Filter |
| IP | Internet Protocol |
| ITU | International Telecommunication Union |
| JSON | JavaScript Object Notation |
| MP4 | MPEG-4 File Format |
| MPD | Media Presentation Description |
| MPEG | Moving Pictures Expert Group |
| MPEG-H | MPEG-H 3D Audio Standard, ISO/IEC 23008-3 (MPEG-H Part 3) |
| MSE | Media Source Extension |
| NGA | Next Generation Audio |
| NMOS | Networked Media Open Standards |
| OBA | Object-based Audio |
| PCM | Pulse Code Modulation |
| RFID | Radio Frequency Identification |
| RTP | Real-Time Transport Protocol |
| UI | User Interface |
| UMCP | Universal Media Composition Protocol |
| XML | Extensible Markup Language |
| VBAP | Vector Base Amplitude Panning |
| VST | Virtual Studio Technology |

# 3. Reference Architecture Specification

## *3.1 Basic components of a radio broadcast workflow*

The typical audio broadcast workflow contains five components, organized in macroblocks:

- Recording
- Pre-production and Mixing
- Radio Studio
- Distribution
- Reception

This workflow, shown in Figure 1, is the result of the long-term experience of broadcasters and content producers and is proven to deliver reliability, efficiency and quality in both the content and technical aspects for the final programme.

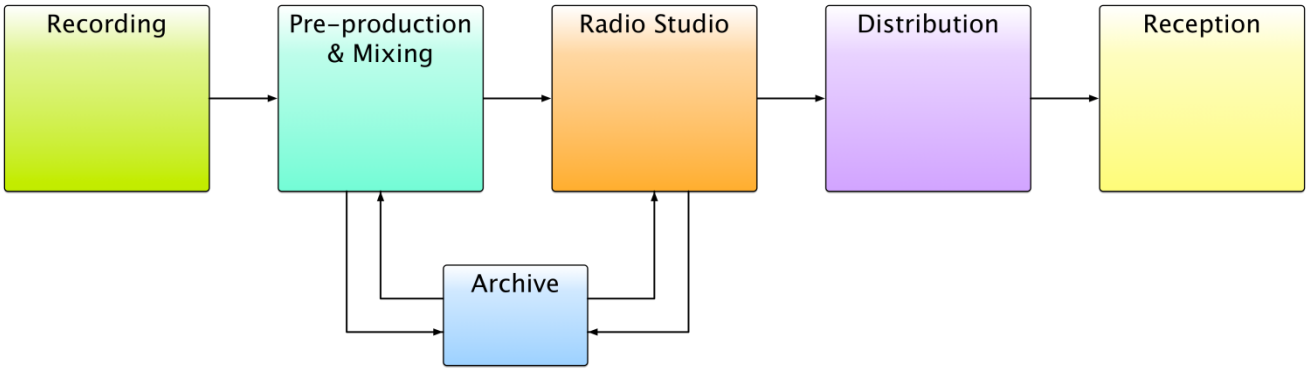



**Figure 1: Basic radio broadcast workflow**

The ORPHEUS project has used this model as the starting point for both pilot implementations and the reference architecture, as it was more practical and efficient than creating a completely new system from scratch.

This approach also helped to demonstrate very early on how feasible it would be to adapt existing broadcast systems to implement object-based broadcasting

However, there are still challenges in delivering the full capability of new object-based media systems and, in order to deliver the full listening experience and achieve the best representation, various features (technical parameters, additional metadata) have to be (automatically) generated and converted, (manually) created and finally injected into the object-based media stream at various points in the traditional workflow.

In the following sections, we describe in detail our considerations, approaches and, occasionally bespoke (single) solutions to resolve these challenges within the limited field of our project.

Thus, inevitably, object-based broadcasting will cause the workflows in existing legacy broadcasting systems to evolve.

To make this eco-system work it appears that we have an example of the classic "the chicken or the egg" problem. What do we need first? Is it production tools or reproduction devices? But unlike other media digitisation challenges the ability to replay object-based media is already widely available, as it is present in the majority of smart phones and personal computers. The distribution infrastructure, in the form of IP networks, is also universal.

This leads us to conclude that enabling production to work with object-based audio is the main challenge.

## 3.2 Reference Architecture

This section describes the actual ORPHEUS reference architecture. It is illustrated at high-level in Figure 2.

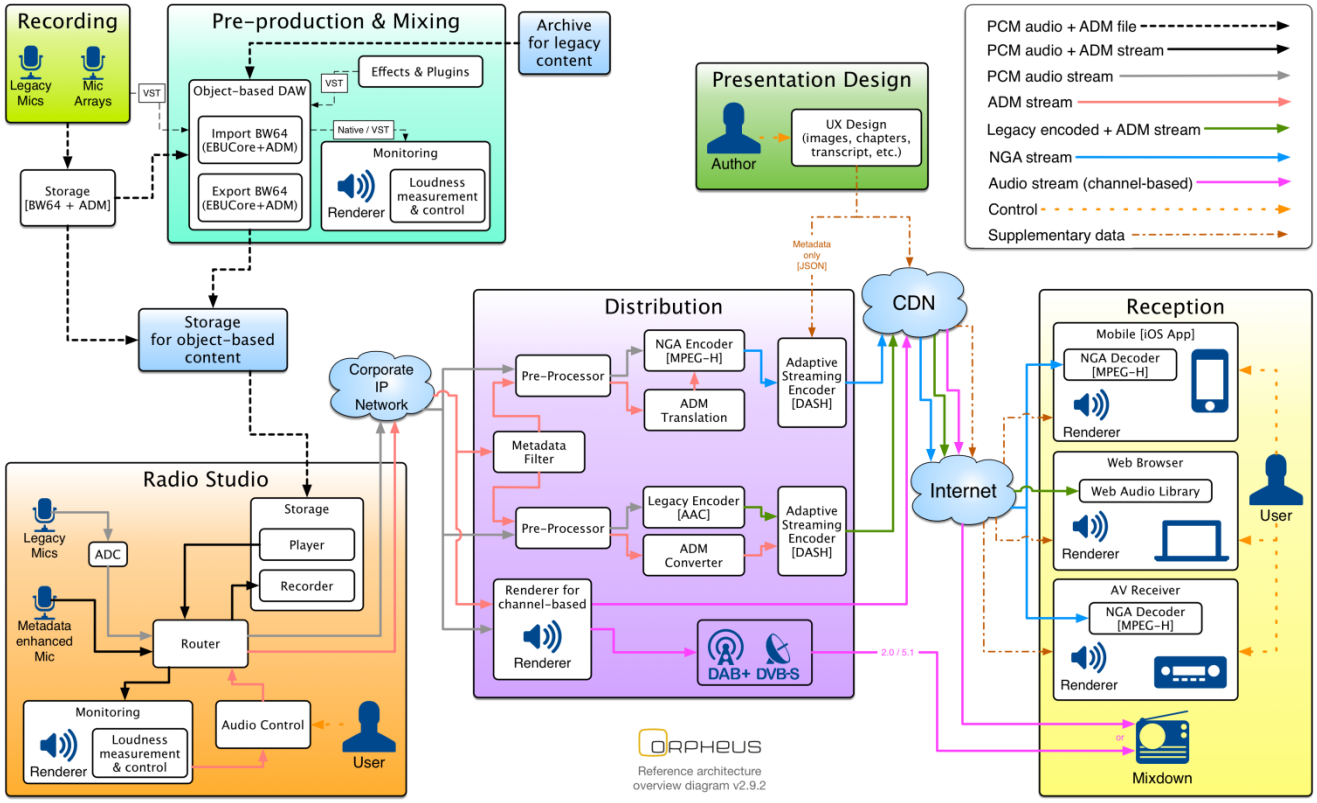



**Figure 2: ORPHEUS Reference Architecture**

The 5 main macroblocks, and the presentation design macroblock, are each described in turn in detail in the following sections.

### 3.2.1 Recording

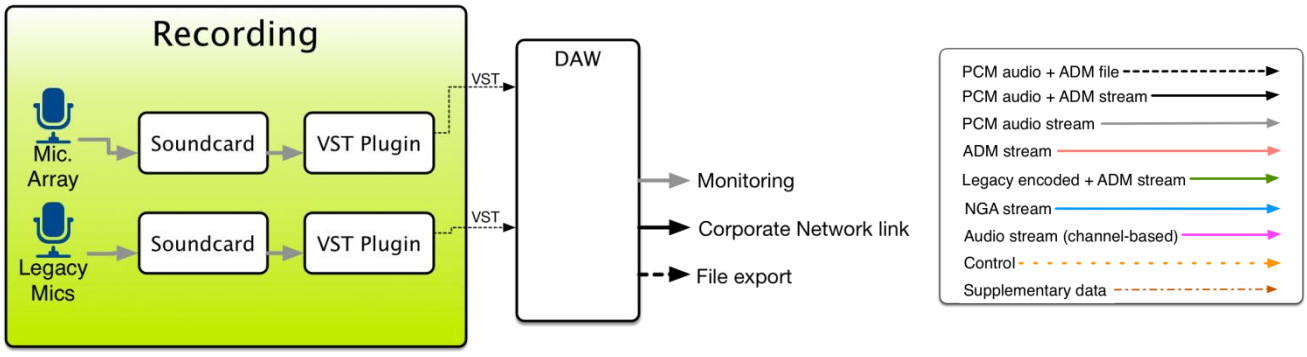



**Figure 3: Detailed Recording macroblock structure of the ORPHEUS reference architecture.**

The purpose of the recording macroblock is to provide the tools and infrastructure required to make recordings for the production of object-based audio content. Currently, object-based audio content, as described by the ADM standard [7], may consist of audio data in the following three formats:

- Pure object-based audio, i.e. mono or multi-channel tracks along with metadata describing the position, width, etc. of the object in the track(s).
- Scene-based audio, i.e. a number of audio tracks describing a scene or sound field. In the scope of this project, this format would be HOA (Higher Order Ambisonics).
- Channel-based audio, i.e. a number of audio tracks corresponding to specific loudspeaker positions. This includes stereo but also more spatialised formats such as 4+7+0[3].

Sound engineers typically record audio using a Digital Audio Workstation (DAW). In this context, recording with specific equipment, such as microphone arrays, or in specific formats, requires the use of audio plug-ins (additional software running within the DAW). The two most frequently used formats for audio plug-ins are Steinberg's VST and Avid's AAX. Note that both VST and AAX are proprietary formats implemented using SDKs that are available from the respective company websites. The VST interface is supported by many more DAWs than the AAX interface, thus the former might be favoured over the latter. However, as Protools (Avid), which supports only the AAX interface, is also widely used in broadcast production environments, it is difficult to make a clear recommendation here.

In the case of a live production, or when the content producers wish to mix the audio signals prior to recording, the recording macroblock should share an interface with the pre-production and mixing macroblock. This interface should typically occur within the DAW, which is used to monitor and/or mix the recorded signals, and possibly add or edit the corresponding metadata. As explained above, the interface would then be either VST or AAX depending on the DAW.

In addition, in the case of non-live productions, the audio data produced by the recording macroblock may be stored for later use. The audio signals should then be stored as BW64 files with ADM metadata. ADM is the standard used for describing object-based audio content in the ORPHEUS project.

***Formats, protocols and Interfaces to other components***

The Recording macroblock has the following output interfaces:

*File formats*

[BW64] Long-form file format for the international exchange of audio programme materials with metadata, Recommendation ITU-R BS.2088.

[ADM] Audio Definition Model, Recommendation ITU-R BS.2076 [7].

*Audio plug-in interface standards*

[VST] Virtual Studio Technology, the Steinberg GmbH interface for audio plug-ins (See the Steinberg website for more information).

[AAX] The Protools (Avid Technology) audio plug-in format. Information about the interface is available from Avid Technology by signing in as a developer.

[3] See Recommendation ITU-R BS.2051: This new nomenclature is used throughout this document

### 3.2.2 Pre-production and Mixing

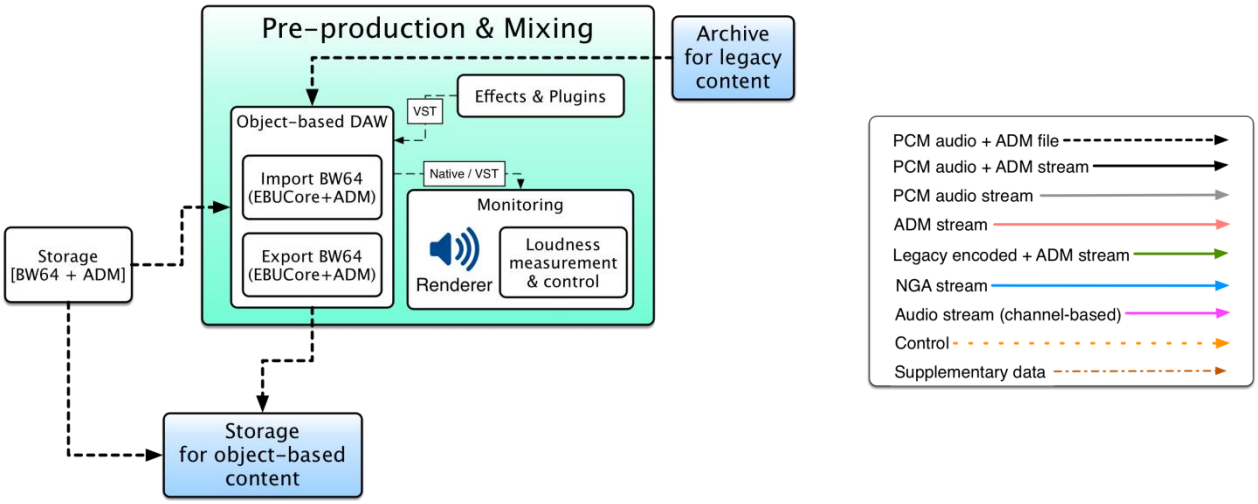



**Figure 4: Detailed Pre-production & Mixing macroblock structure of the ORPHEUS reference architecture**

The purpose of the pre-production and mixing macroblock is to deliver tools for editing existing object-based content or creating such content from legacy audio material or other sources. The core of this macroblock is the object-based DAW, which is extended by several tools and workflows to import, edit, monitor and export object-based content. The DAW interacts with the components presented in the section below and illustrated in Figure 4.

***Description of DAW components:***

The following main components of the object-based DAW were identified:

*Recording*

The recording of object-based content may be done independently from the DAW. In this case the recorded content is transferred in form of multichannel BW64 files with ADM metadata. Depending on the established broadcast architecture, these files might be stored in temporary **ADM-enabled storage** or directly in the **storage for object-based content**.

Additionally, there is the option to record object-based content directly within the DAW. This implies further technical requirements for the DAW (e.g. 32 input channels for the microphone array processing that could be used in the recording macroblock).

*Storage for object-based content*

The DAW needs to be able to import and export object-based content in the form of uncompressed multichannel BW64 files with ADM metadata. This involves **decoding** and **encoding** these files including their metadata. Editing existing ADM metadata or additional metadata required for broadcasting may be implemented directly in the DAW or by stand-alone tools.

*Archive for legacy content*

For integration into an established broadcasting architecture, access to existing content is very important. For this purpose, the DAW can at least convert legacy content to object-based content by adding the required object-based metadata to it manually. This process could be automated partially.

*Effects and plug-ins*

A typical use for a DAW is the application of effects or plug-ins onto audio tracks. Basically, applying an effect consists of taking a given sound signal and changing it somehow. Common effects

used in audio production are filtering (changing the frequency content of a sound) or dynamics processing (a limiter or compressor, to change the dynamic range of an audio track). Plug-ins are extensions for the DAW to be used with audio tracks.

Usually effects and plug-ins can be applied to audio content within the DAW. Due to the nature of the ADM format there are some limitations to consider here. For example, there might be no final mastering stage for a certain channel format, and effects can be applied to individual audio objects only.

For the scope of ORPHEUS, it was decided that any relevant plug-ins should be made available as VST 2 plug-ins. This also included format conversions e.g. converting 3D microphone input to HOA format or converting HOA format to channel-based surround formats. Plug-ins for these use cases already exist and are delivered e.g. by *b<>com*.

A special use case is applying reverberation to the audio signal. Currently reverberation could only be included as a fixed part of the individual objects or as an additional audio object, which partially limits the possibilities of ADM.

To deal with 3D spatial audio, the panning engine of the DAW needs to provide a suitable interface for handling 3D automation curves and to convert them from and to the ADM objects as well as to the renderer plug-in.

*Monitoring*

For previewing the object-based content and simulating the audience experience, it is important to allow monitoring with different loudspeaker setups and binaurally. For this purpose, an object based renderer needs to be implemented in the DAW. ITU-R Working Party 6C is currently in the process of standardizing one or multiple options for a production renderer. For ORPHEUS, however, the MPEG-H renderer was chosen, since the ITU-R standardization is still ongoing. Moreover, the EBU has published Tech 3388 [EBU ADM Renderer], which could also be used for the production of next-generation audio (NGA) programmes.

An essential part of the monitoring is loudness measurement and control. This may be either supported by additional VST plugins in the DAW or implemented as part of the rendering solution.

In addition to pure audio monitoring, other parts of the user experience may be simulated within the DAW, e.g. transport control, projects with variable length or exchanging language specific objects.

*Formats, protocols and Interfaces to other components*

| | |
|---|---|
| [BW64 + ADM] | Audio Definition Model (ADM): BS.2076-1, June 2017, ITU. |
| [WAV][AAC][MP3] | Or other common audio formats to be imported from legacy content archives |
| [VST] | Virtual Studio Technology, the Steinberg GmbH interface for audio plug-ins (See the Steinberg website for more information). |
| [EBU ADM Renderer] | EBU ADM Renderer (EAR): Tech 3388, March 2018, EBU |

### 3.2.3 Radio Studio

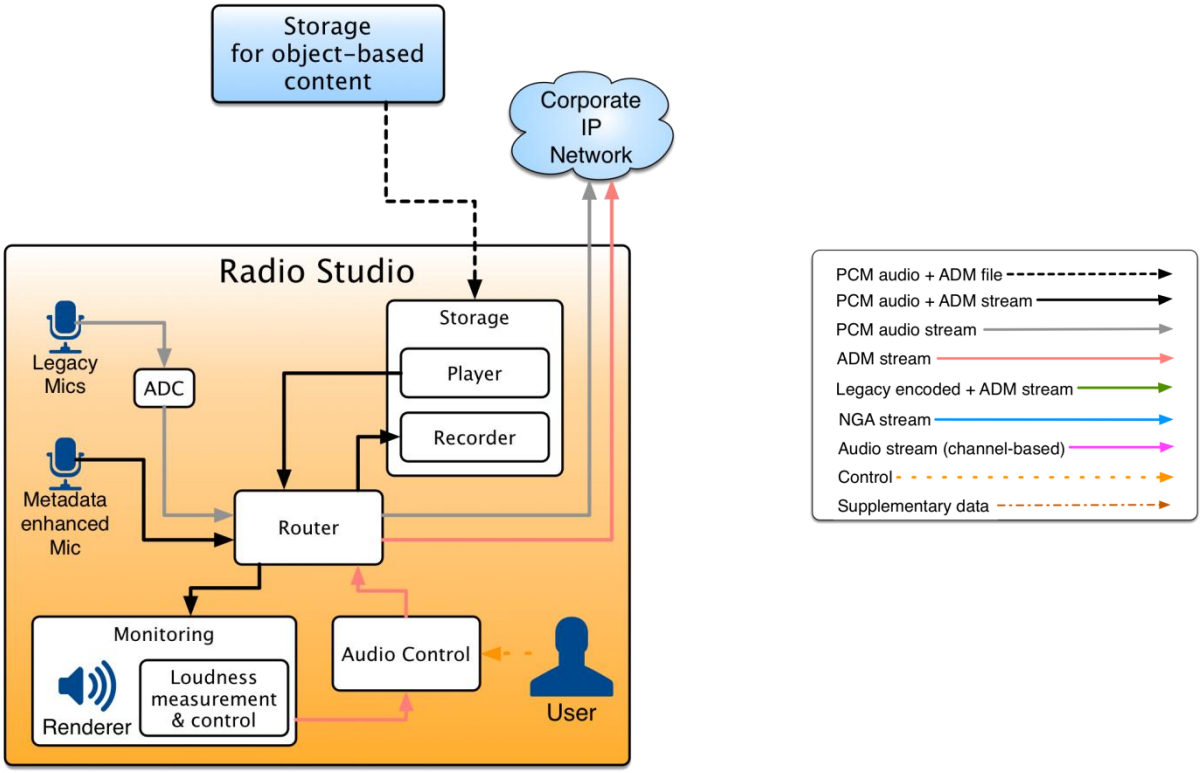



**Figure 5: Detailed Radio Studio macroblock structure of the ORPHEUS reference architecture**

There are three main aims to the radio studio macroblock: capture and control, monitoring, and record/replay. Each of these is explained below, with detail about the functionality within each. We then describe the interfaces used to communicate between components.

***Capture and control***

In the studio macroblock, we want to be able to capture audio sources, and describe how these and other audio sources should be combined into a single programme, or 'composition'. In a traditional studio, these audio sources would be mixed together to produce a new audio source, however with the object-based approach, the audio sources remain separate and we generate metadata to describe how they should be combined.

In describing a composition, the following information must be captured, stored, and transmitted alongside the audio:

- **Sources**: Which audio sources are parts of the composition, and to which audio source the metadata relates.
- **Gain and position**: How much gain should be applied to each audio signal, and where the audio should be panned within the auditory scene.
- **Labelling**: A description of what the audio source is. This will vary based on the context of how the audio is used. For example, the labelling could describe whether the audio source is a foreground or background sound, or identify a person that is speaking. Some of this information could be automatically gained using devices that identify themselves and their location (e.g. a networked microphone, enabled with an RFID reader)

We have chosen to use the NMOS specifications for our radio studio. This provides open standards for mechanisms that handle identification, timing, discovery and registration. We use the Audio Definition Model (ADM) to describe audio metadata such as gain, position and labelling.

*Audio control interface*

To capture and transmit this control data, there must be a user interface which the engineer or producer can use to achieve this. The features of the user interface must include the following:

- **Creating a new composition**: Generating a blank programme that will be populated with audio objects.
- **Adding objects**: Selecting a new audio source and including it in the composition.
- **Controlling gain/position**: Being able to set the gain/position of each object dynamically.
- **Labelling**: Setting labels for each audio object appropriate to the context of the use case.

Traditionally, the audio control interface would be a mixing desk, made up of a series of faders and knobs. However, to be able to capture the rich metadata required for an object-based production, it may be necessary to replace the mixing desk with a graphical user interface, or at the very least, use a combination of physical controller and graphical interface.

*Pre-production interface*

In a live workflow, some of this data may have to be entered on-the-fly, such as gain and position, however much of the data can be prepared in advance. For example, the identity of contributors is often known in advance, so this information should be captured in pre-production. This pre-production information could be captured using a specialised interface designed for the producer, who generally looks after the editorial side of the programme. Examples of data to be captured using this interface include:

- **Programme title/description**: An overall view of the composition
- **Running order**: A sequence of chapters or scenes in the programme, with descriptions for each
- **Contributors**: A list of people who are involved in the production, including producer, engineer, writer, presenter and guests.
- **External Links**: This could include URLs related to the content, or a link to a related image.

Once pre-production information has been captured, then its use must be triggered at the appropriate moment. This could be done manually through the audio control interface with the engineer pressing a button. Otherwise, it could be done automatically using other inputs such as ID badge readers, or using signal processing to work out when a microphone is being used.

*Metadata transmission*

Methods for the transport of audio over networks is well specified (e.g. AES67) however work on defining techniques for transporting audio metadata is in its early stages. We are investigating using the Universal Media Composition Protocol (UMCP), which is being developed at the BBC for use in media metadata transport.

UMCP provides a way to describe ‘compositions’, i.e. how various NMOS-described sources can be combined into a single experience. This protocol provides a flexible and sustainable way to link metadata to audio streams. It can either be carried over WebSocket connections, or using RTP packets. UMCP uses a server model to store incoming compositions and distribute them to any receivers that have subscribed to updates.

***Monitoring***

When producing an audio experience, it is important to be able to monitor the audio output and ensure it is complying with the appropriate technical and editorial standards. This can be done using two methods: first by listening to a rendered version of the output, and secondly – only for

technical compliance - by using metering to visually monitor each object and the output.

*Audio monitoring*

A renderer is a system that takes an object-based audio composition and generates a set of output signals to drive loudspeakers or headphones. For monitoring the output of a production in a radio studio, this typically involves rendering to a set of loudspeakers so that the engineer and producer can listen to the output. Alternatively, it could be rendered to headphones for individual monitoring.

A variety of rendering systems are available, but it is up to each organisation to select the one most appropriate to them. There are currently efforts in the ITU-R to standardise a 'production renderer', however this extends beyond the timeline of the ORPHEUS pilot. For the pilot, we used a VBAP-based renderer, starting with the BBC's own implementation, succeeded by the MPEG-H rendering system. The EBU's recently published Tech 3388 [EBU ADM Renderer, see § 3.2.2] could also be used for monitoring.

*Audio metering*

Metering involves processing an audio source using an algorithm to calculate the level of some property of the signal. The levels that we use in ORPHEUS are loudness and true peak, as described by EBU R128. Each audio object is routed through a processing node to calculate the levels, and these can then be displayed on the audio control interface. This allows the engineer to monitor the audio signals of each object. The output of the studio renderer is also metered to ensure the overall output levels are appropriate.

***Recording/replay***

The ability to record and replay material is an essential aspect of audio production. However, current tools only offer a way to record the audio data. For object-based production, we need to be able to record and replay both audio and metadata. To achieve this, we are using a 'sequence store' in IP Studio, which records audio and metadata flows. These are recorded into a database along with the timestamps of each grain in the flow.

To interact with the sequence store, we must use an API. For the ORPHEUS pilot, we are using a BBC developed interface called the 'Media Access API'. This allows systems to trigger the recording and replaying of streams.

The API to the sequence store can also be expanded to allow for importing/exporting of various formats. For the ORPHEUS pilot, we have implemented an import script, which can read BW64+ADM files, convert them to a stream of audio and metadata, then import that into the store.

***Formats, protocols and Interfaces to other components***

The Radio Studio macroblock has the following output interfaces:

*Live output to Distribution*

This macroblock exports an object-based stream as a UMCP composition containing ADM metadata of NMOS audio sources. An overview of these protocols is described below:

**[NMOS]** Networked Media Open Standards are a set of open specifications, which define four important aspects of interfacing object-based audio over IP:

1. **Identity** – How to identify each individual audio source and stream of data ('flow') from those sources.

2. **Timing** – How to define when audio was sampled, when metadata changed, or when events occurred.

3. **Discovery and registration** – How devices announce themselves as being available, and advertise what streams they can offer.

4. **Routing** – How streams get routed between devices on the network, and how devices can request to receive streams.

The NMOS specification is available at http://nmos.tv

**[ADM]** Audio Definition Model is used as a data model as a standard way to describe a sound scene made up of audio objects. ADM has been used in two ways:

1. in combination with BW64 to describe the contents of object-based audio files

2. as a data model to describe audio parameters over UMCP for streaming of object-based audio.

The ADM specification is described by Recommendation ITU-R BS.2076 "Audio Definition Model" [7].

**[UMCP]** Universal Media Composition Protocol is used to link the ADM metadata to NMOS-described audio sources. Currently (January 2018), ADM is used in a file format and can only describe audio sources as channels within the file. In a live production, the audio objects are distributed on a network so cannot be described as 'channel 1', for example. UMCP solves this problem. UMCP links into the NMOS specifications we are using to identify audio sources, and to describe the timing of events. Although a UMCP stream could be sent directly from a transmitter to a receiver, it is normally routed through an API that records the stream and distributes it to any receiver that has subscribed to it. UMCP is currently being developed into an open specification for publication.

For further information about subsets or specific profiles of the identified interfaces used, see ORPHEUS Deliverable D4.2 [9].

### 3.2.4 Presentation Design

A key motivation for object-based broadcasting is the desire to make a transition from monolithic 'one-size-fits-all' streams that are fixed in both content and presentation, to programmes that can be personalised to a listener's interests, wishes, and playback environment.

Object-based broadcasting can enable a wide variety of novel interactions with the audio broadcast content, both in the determination of what is rendered and how it is rendered. For a particular programme and a particular user or audience, a specific subset of the possible interaction features may be selected. To make this interaction simple, understandable and effective, specific user interfaces need to be designed. As the design of these user interfaces can have a profound effect on the total experience, it is a component to be considered already during creative and editorial processes.

The presentation design macroblock, shown in Figure 6 becomes an important part of this new workflow.

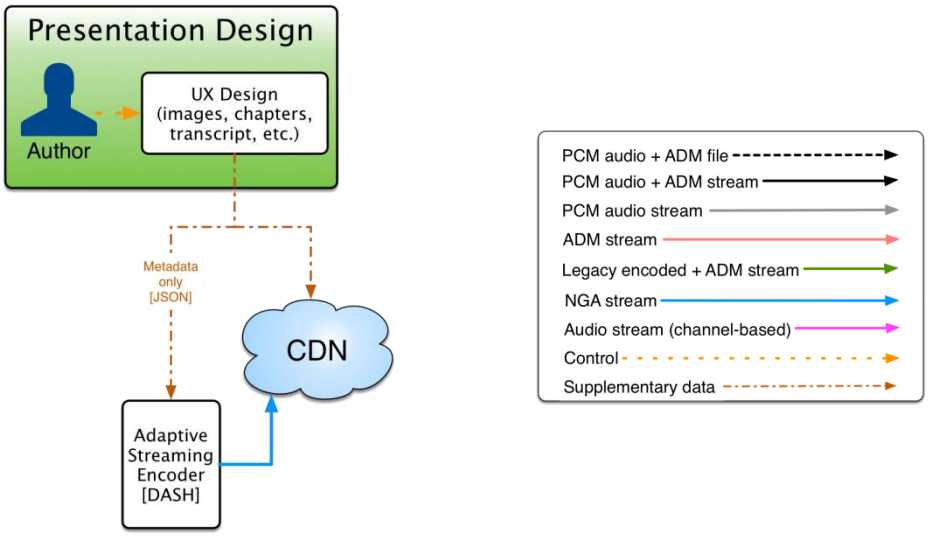



**Figure 6: Detailed Presentation Design macroblock structure of the ORPHEUS reference architecture**

Besides interaction with the audio objects in the broadcast, such as language selection, level adjustments, non-linear playback and spatialisation, OBA is very well suited to be complemented with additional information and metadata. This might be complementary text and images relating to narrated text and speakers, multi-lingual programme information or content-related tags.

***Formats, protocols and Interfaces to other components***

The formats and protocols that can be attached to an object-based audio stream are determined mainly by the presentation that the broadcaster wants to provide: for a highly interactive program in a browser, standard web technologies like HTML can be used; if the goal is to just add some extra labels or textual information, a stream of inline JSON data within MPEG-DASH will suffice.

### 3.2.5 Distribution

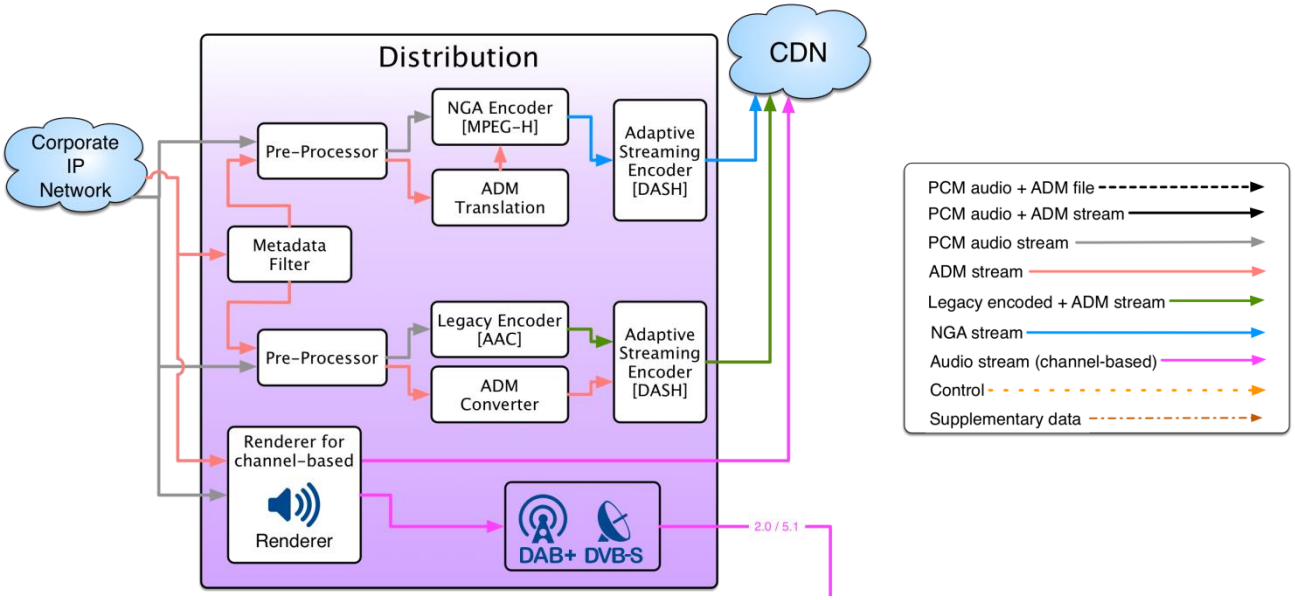



**Figure 7: Detailed Distribution macroblock structure of the ORPHEUS reference architecture**

#### 3.2.5.1 Scope and Functionality

This macroblock contains the modules and tools needed for the distribution of object-based audio from the broadcaster to the end user. It converts the production format used within the broadcast infrastructure (AES67, BW64, ADM) into a more efficient format that is suitable for transport over transmission channels and reproduction in receiver devices (AAC, MPEG-H, DASH). The main distribution channel is the Internet (IP, TCP, HTTP) but, for backward compatibility, legacy systems (DAB+, DVB-S, Shoutcast) are considered in the Reference Architecture.

The distribution macroblock receives its input from the studio macroblock via a private IP network as illustrated in Figure 2. For live production, the object-based audio is received as PCM via AES67 plus an ADM-based metadata stream. Both these macroblocks use UMCP as the underlying protocol to establish and control data streams between each other (including audio and metadata). As the input interfaces and UMCP are already described in § 3.2.3, this section focuses on the modules within the distribution macroblock and its output interfaces.

1. **Internet Distribution**: The ORPHEUS Reference Architecture includes two options for distribution over the Internet: An AAC-based distribution to HTML5 browsers, and another for clients which support MPEG-H 3D Audio (iOS-App, AV-Receiver). Both paths are made available via the public Internet and are based on HTTP/TCP/IP as the underlying transport and network protocols of the World Wide Web. In addition, both use DASH (the most widely adopted streaming protocol today). As a consequence, a Content Distribution Network (CDN) can be used for scaling the service to many users and several geographical regions. As the usage of a CDN is transparent to the services defined in this document it is not covered in more detail.

2. **Distribution to legacy devices**: Though legacy devices, such as DAB+ and/or DVB receivers will never support the full functionality of object-based audio, they are important for backward compatibility to mass market legacy broadcasting systems. To enable this additional distribution path, it is necessary to render object-based audio to channel-based versions, e.g. as a simultaneous downmix into 2, 0+5+0 surround, and binaural. In addition, alternative language versions can be mapped into audio sub-channels and a subset of the audio-metadata can be mapped into existing metadata models of legacy systems. A similar functionality is required when emission via a classic Internet radio system, such as Shoutcast, is required.

3. **Filtering of private metadata**: For object-based audio, specific metadata has to be transmitted to the end-users as part of the distribution. However, not all metadata available within production is intended for the end-user. Hence, any information that is ‘private’ or used solely for internal purposes needs to be removed before distribution.

4. **Pre-Processing for Distribution**: Depending on the capabilities of the distribution format (e.g. MPEG-H), the produced/archived audio content and metadata may need to be pre-processed or converted. For example, the ADM metadata might need to be either translated into a different metadata representation or a new/modified ADM metadata stream needs to be created for emission. This process may also include a reduction of the total number of objects or a combination of several objects into one audio track or channel bed. Such a conversion process typically translates a more generic ADM metadata set into a more constrained *ADM-Profile*, such that conversion into MPEG-H or other NGA codecs becomes more predictable.

5. **Conversion of ADM to MPEG-H Metadata**: Though the pre-processor can significantly reduce the complexity of ADM-based content, there still needs to be a final conversion from ADM into the metadata model of the emission codec, e.g. MPEG-H. This step must

be as transparent and predictable as possible and should be a direct mapping of metadata elements. This, however, can only be achieved if a rather constrained and well-defined subset of ADM is used as input. To address this requirement, Fraunhofer has specified an "ADM Broadcast Emission Profile" which it is making available to the ITU and companies that are working on current products and deployments of NGA [ADM-EP]. This FHG document specifies a subset of ADM in such a way that it can be used as a flexible, transparent, and standardized way for controlling low-bitrate audio codecs for Next Generation Audio.

#### 3.2.5.2 Output Interface Specification

In this section we describe the two main distribution paths to the end user, which are addressed in the ORPHEUS Reference Architecture. After providing a textual overview, we list the relevant specifications that define the output interface of the distribution macroblock. We focus on the audio-related distribution but note that additional metadata is typically needed to achieve a good overall user experience, e.g. additional program information or text and images etc. The definition and transport of such additional metadata is described in § 3.2.4.

***Distribution to HTML5 Browsers***

In order to reach a broad audience, it is desirable to support commonly-available browsers as reception devices. However, as current browsers do not yet support Next Generation Audio (NGA) audio codecs such as MPEG-H, it is necessary to implement the required functionality as a *Web Application* based on JavaScript and other available browser APIs.

Web applications for object-based audio have already been demonstrated successfully and use HTML, CSS, and JavaScript for implementation. When the user visits the web site of a broadcaster, the web application is automatically downloaded and executed by the browser as part of the web page. It then downloads the Media Presentation Description (MPD) to start a DASH stream. The MPD links to the actual content, which is then downloaded as a series of fragmented MPEG-4 (fMP4) file segment. For decoding the fMP4 segments there are two approaches:

1) For audio programmes of up to 8 mono objects the audio stream can be decoded using the HTML5 media elements in the browser.
2) For programmes with more than 8 channels, multiple streams must be used currently as the browser decoders support only up to 8 channels per stream. As no reliable synchronisation between multiple HTML5 media element decoders can be achieved, the WebAudio API is used to decode the media into AudioSourceNode objects.

These can then be scheduled for playback with sample accuracy and linked to a timeline. Finally, the WebAudio API is used to render the audio using the JavaScript rendering engine, e.g. to generate a binaural stereo signal from multiple audio objects. In order to do so, the web application also needs the audio metadata, e.g. the position of objects in 3D space. Those are also streamed via DASH as file segments including ADM metadata, which can either be transmitted as XML or encoded as JSON. Though the exact implementation of the browser-based web application is out of scope for the distribution macroblock, the named browser APIs implicitly define the required output interfaces (JavaScript, CSS, HTML5, WebAudio) and are therefore mentioned here.

It is worth noting that the exact definition of the streaming protocol does not have to be known as long as the client implementation itself is downloaded as "mobile code" along with the data. For example, the exact protocol of how audio metadata is represented and transmitted does not have to be defined as long as the JavaScript-based web application knows how to receive, parse, and interpret it correctly in order to drive the WebAudio API for rendering. In this sense, the distribution macroblock only has to make sure that the encoded fMP4 segments are compatible with

the HTML5 capabilities of the browser and that the web application provided uses the WebAudio API correctly.

Figure 2 illustrates how the following functionality is implemented in the macroblock for the distribution path to browsers: First, the ADM metadata has to be received from the studio macroblock and interpreted, e.g. to configure the AAC encoder with the appropriate number of objects. The AES67 stream (or BW64 file) is received and the PCM audio is fed into a multi-channel AAC encoder. The compressed AAC bit-stream is encapsulated into fMP4 file segments and stored on a DASH server which is also responsible for generating the MPD. In addition, the file segments with XML-or JSON-encoded ADM metadata are generated, which are also streamed via DASH as a parallel data stream.

The distribution-macroblock has the following output interfaces to the reception macroblock when streaming to HTML5 browsers:

| | |
|---|---|
| [HTML5] | HTML5, A vocabulary and associated APIs for HTML and XHTML, W3C Recommendation 28 October 2014, https://www.w3.org/TR/html5/ |
| [CSS] | Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, W3C Recommendation 07 June 2011, https://www.w3.org/TR/CSS2/ |
| [JavaScript] | ECMA-262, ECMAScript 2016 Language Specification, 7th Edition / June 2016, http://www.ecma-international.org/publications/standards/Ecma-262.htm |
| [WebAudio] | Web Audio API, W3C Working Draft 08 December 2015, https://www.w3.org/TR/webaudio/ |
| [AAC] | ISO/IEC IS 14496-3:2009, Information technology - Coding of audio-visual objects - Part 3: Audio |
| [DASH] | ISO/IEC 23009-1, Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment format |

#### *Distribution to Clients supporting MPEG-H 3D Audio*

Though the browser-based distribution is a flexible solution, that is immediately deployable, it has several shortcomings compared to a "true" NGA codec such as MPEG-H. First, the implementation in JavaScript is less efficient than a native C implementation and therefore the computational complexity can become a problem, especially on mobile phones. Secondly, rendering via the WebAudio API has the inherent limitation that audio content is not protected but "in the clear", which can become a problem for premium content. Furthermore, MPEG-H supports not only objects but a flexible combination of channel-, object-, and scene-based (HOA) audio formats – all with a significantly-improved coding efficiency compared to an AAC-based approach. Finally, MPEG-H provides a well-defined behaviour based on an open and interoperable standard, which can reduce the implementation and maintenance effort for broadcasters.

Because MPEG-H is an open standard and is already adopted in several application standards, the output interfaces for this path are well-defined and documented. In addition to the actual MPEG-H 3D Audio standard, which defines the overall audio codec [MPEG-H], it is typically required to define a subset for a specific application. The ORPHEUS Reference Architecture follows the Profile and Level that has been defined in ATSC 3.0 and DVB, namely the *Low Complexity Profile at Level 3* [ATSC-3]. Those application standards also include further definitions and clarifications on the usage of MPEG-H, which also apply to the Reference Architecture. In addition to the audio codec, the output interface also covers the usage of MPEG-DASH [DASH], which itself is a flexible standard that needs further profiling and clarification. Here, the ORPHEUS Reference Architecture follows the recommendations of the DASH Industry Forum (DASH-IF), which not only published its

Implementation Guidelines but also provides test data for interoperability testing [DASH-IF].

Also in Figure 2 we see how the following functionality is implemented in the macroblock for the distribution path to end-user clients that support MPEG-H 3D Audio: First, the ADM metadata has to be received from the studio macroblock and interpreted, e.g. to configure the MPEG-H encoder with the appropriate number of objects or channels. This requires a conversion of metadata from ADM to MPEG-H. The AES67 stream (or BW64 file) is received and the PCM audio is fed into the MPEG-H encoder. In addition, dynamic ADM metadata (if any) has to be received, converted and fed into the MPEG-H encoder. The compressed MPEG-H bit-stream is encapsulated into fMP4 file segments and stored on a DASH server who is also responsible for generating the MPD.

Any client supporting MPEG-H and DASH according to the output interface defined above can receive object-based audio from the distribution macroblock. Within ORPHEUS two client implementations have been implemented and thus verified feasibility: An iOS-App to be executed on an iPhone and an AV receiver for immersive reproduction at best quality.

The distribution macroblock has the following output interfaces to the reception macroblock when streaming to clients supporting MPEG-H 3D Audio:

[MPEG-H] ISO/IEC 23008-3:2015, 2015, Information technology - High efficiency coding and media delivery in heterogeneous environments - Part 3: 3D audio

[ATSC-3] A/342 Part 3:2017, MPEG-H System, March 2017, http://atsc.org/wp-content/uploads/2017/03/A342-3-2017-MPEG-H-System-1.pdf

[DASH] ISO/IEC 23009-1, Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment format

[DASH-IF] Guidelines for Implementation: DASH-IF Interoperability Points, Version 4.0, DASH Industry Forum, December 12, 2016

[ADM-EP] ADM Broadcast Emission Profile. Available on request from Fraunhofer, please contact amm-info@iis.fraunhofer.de

For further information about used subsets or specific profiles of the identified interfaces, see ORPHEUS Deliverable D4.2 [9].

### 3.2.6 Reception

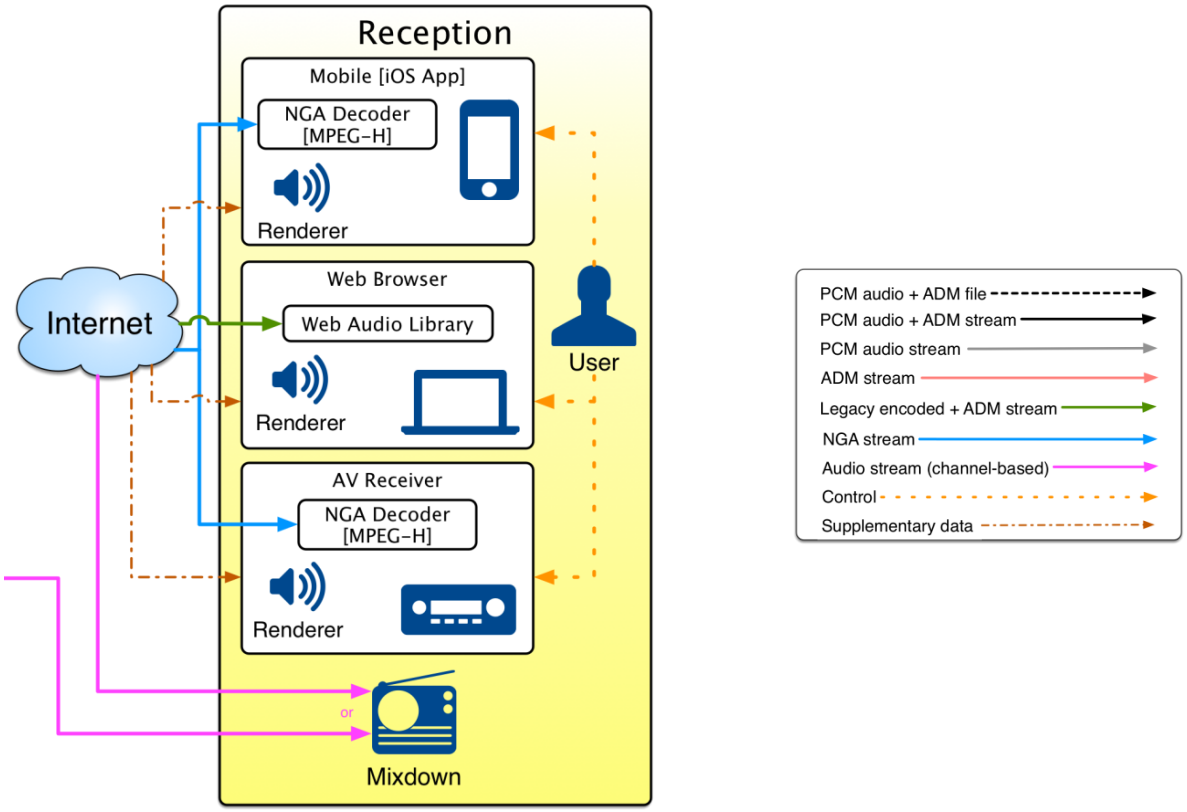



**Figure 8: Detailed Reception macroblock structure of the ORPHEUS reference architecture**

The purpose of this macroblock is to provide solutions for the reception, personalisation and reproduction of object-based audio content for the end-users. Hardware and software solutions are considered in order to meet different segments of the consumer electronics market, as well as different end-user listening habits and audio content consumption contexts. These solutions differ in terms of rendering capabilities, according to the available network bandwidth, number of audio output channels and processing power. They also differ in terms of user interface and proposed personalisation/interactivity features since they are not addressing the same listening contexts (e.g. domestic use vs mobility) and are equipped with different sensors and input devices (screens and keyboard, touch screens, built-in microphones, GPS sensors etc.).

All solutions are comprised of two major components: a **decoding/rendering module** and a **personalisation module**. The decoding/rendering module receives and decodes the compressed audio streams and adapts the play-out of the content according to their type (channel-, scene- (HOA) and object-based) and to the end-user environment's setup (e.g. binaural rendering over headphones or over conventional or *ad-hoc* multi-channel loudspeaker setups).

The personalisation module provides the means for displaying and altering received metadata related either to mixing/spatial features as well as semantic properties. For instance, the user interface provides the means for adapting the audio quality to the rendering setup or to the listening context (e.g. 3D immersive audio over headphones or loudspeakers, more conventional play-back, compression level). The user interface displays "content metadata" such as music title, performer, composer, author, names of panellists as well as live text transcription. When applicable, it also provides the means for selecting the preferred language.

To provide the best possible user experience, some of the personalisation features need to go beyond traditional content authoring. A radio drama, for example, can be enriched with a special user interface as was done in Pilot 1 (see Section 2 of ORPHEUS project deliverable "Interim Pilot

Progress Report" [11]). Additional images, the chapterisation, transcripts and other user features are further examples and require an additional authoring & creation process, a *Presentation Design*, as introduced in this reference architecture in § 3.2.4.

Optionally, environmental adaptation is proposed, with inputs from environment sensors (head-tracking for binaural rendering, compensation for background noise, rendering setup alignment/equalisation).

The general features of the decoding/rendering module and user interface are described hereafter for each hardware or software solution.

### *Mobile devices*

Smartphones and tablets are the most dominant platforms for future media consumption. On mobile devices, it is common to install custom apps instead of using the browser as it allows for the integration of more efficient renderers. An iOS app to be executed on an iPhone is being designed and implemented by the company Elephantcandy to highlight and exploit innovative features of object-based broadcasting, with specific attention paid to the personalisation of rendering and reception, based on user preferences and environmental situations. Mobile devices can be connected to a wide range of types of reproduction hardware, ranging from headphones and internal speakers to (wireless) loudspeakers and multi-channel surround systems. Therefore, a mobile object-based audio app should also support multiple audio output formats, such as mono, stereo, binaural and 0+5+0 surround sound.

The reception macroblock for the mobile phone device is depicted in Figure 8. It receives MPEG-H 3D audio streams over DASH (See § 2.2.5). The decoder/renderer of the MPEG-H client implements the Low Complexity Profile Level 3 of the MPEG-H standard. In its current implementation, it is limited to a maximum of 16 simultaneously active (that is, to be rendered) signals out of a total of up to 32 channels, objects, and/or HOA signals in the bit-stream as specified in the standard.

For a typically individual device, personalisation features are important. Object-based audio allows for personalisation at the level of audio rendering, with features such as foreground/background balance adjustment to increase the intelligibility or clarity of the sound, dynamic range compression to compensate for background noise, or customised spatialisation settings.

In addition to user determination of how the audio is rendered, the rich metadata in an object-based audio stream can also give the listener increased control of what is played. Sections can be skipped or relayed through interaction with "points of interest" or textual transcripts. Automatic selection, based on user-selected criteria, can be used to achieve variable-length playback, where a radio programme is reassembled on the mobile device in a personalised way.

As not all metadata required for a particular app or broadcast may be included in the MPEG-H stream, additional JSON streams can be streamed in parallel, possibly linking to resources outside the object-based audio architecture specified in this document.

### *Audio hardware devices*

The hardware receiver designed and built by Trinnov is a CPU-based device. The audio processing is a closed software library running on an embedded computer. Some third-party libraries such as audio decoders may be included as independent modules, linked to the main library and run from the main signal processing function. The hardware receiver is designed for luxury "home cinema" rooms. Thus, the typical reproduction systems that will be connected to it are multi-loudspeaker set-ups. To elicit the sensation of immersion, most common setups have several layers of loudspeakers, including some elevated ones. This is a chance to render, in a very accurate way, spatialised audio objects that can be static or moving around the 3D space.

The reception macroblock for the audio-video device is depicted in Figure 8. It receives MPEG-H 3D audio streams over DASH (see § 2.2.5). As for the mobile device, the decoder/renderer of the MPEG-H client implements the Low Complexity Profile Level 3 of the MPEG-H standard. In its current implementation, it is limited to a maximum of 16 simultaneously active (that is, to be rendered) signals out of a total of up to 32 channels, objects, and/or HOA signals in the bit-stream as specified in the standard.

The main audio playback formats supported by the renderer are binaural over headphones, VBAP or HOA decoding over conventional loudspeakers layouts, as well as ad-hoc 2D or 3D loudspeakers layouts (up to a maximum of 32 output channels).

The loudspeaker layouts differ most of the time from the recommendations in terms of speaker positioning, to a greater or lesser extent. Thus, the rendering has to deal with improper loudspeaker layouts, including misplaced or even missing loudspeakers, and the content has to be adapted to these configurations. This adaptation is done in 2 steps:

- First, the closest configuration is selected by the user, and the decoding/rendering module provides *ad hoc* contents for the channels, HOA scenes and/or objects;
- Secondly, the personalisation module, after an acoustic calibration of the system, performs an optimisation to compensate as much as possible the deviations from the reference, starting from the level and delay adjustments for all the output drivers.

The personalisation module can also compensate for the acoustic dispersion of the loudspeakers, since they are rarely all similar in real-life installations: people usually favour higher quality loudspeakers for front channels and lower cost ones for surround or elevation ones. Furthermore, multiway loudspeakers are often used for the front channels whereas they are hardly ever used for other positions. This disparity can cause bad rendering, especially of objects that should be panned between loudspeakers of different types or quality.

When this environmental adaptation is performed, the content personalisation is also performed, as for the mobile app described above: foreground/background balance adjustment to increase intelligibility, dynamic range compression to compensate for background noise, or customised spatialisation settings are offered to the user in the interface.

#### *Web browsers*

Today, an HTML5 capable web browser can be considered as the most universally available reception system. Its ubiquity means that a very large audience can be exposed to object-based broadcasting without any extra effort required (such as downloading plug-ins, installing apps, configuring loudspeaker layouts etc.). Browsers can be used on devices with a large variety of reproduction hardware, ranging from headphones and internal loudspeakers to (wireless) loudspeakers and multi-channel surround systems. Therefore, the web renderer should also support multiple audio output formats, such as mono, stereo, binaural and 0+5+0 surround sound. At present, web browsers do not readily support object-based audio rendering. The ORPHEUS consortium, however, worked on the implementation of a rendering system based on JavaScript and the Web Audio API (WAA). While a native solution will not be available for all browsers immediately, growing support for and maturity of the Web Audio API makes this a promising option.

The reception macroblock, which contains the Web Browser, is depicted on Figure 8. It receives AAC audio streams and JSON-encoded ADM metadata over DASH (See § 2.2.5). The DASH receiver decodes the encapsulated AAC bit-streams using the Web Audio API. The multiple audio objects are rendered according to transmitted ADM metadata using the WebAudio API.

The Web Audio API[4], developed within the World Wide Web Consortium (W3C) is a platform dedicated to audio processing within web applications. It provides specifications and descriptions of a high-level JavaScript (JS) API for different audio tasks such as mixing, processing, and filtering. The WAA provides a number of native audio node objects (e.g. *AudioBufferSourceNode*, *GainNode*, *PannerNode*, *ConvolverNode*, *AudioDestinationNode*…) that can be connected together to form an audio graph (see Figure 9).

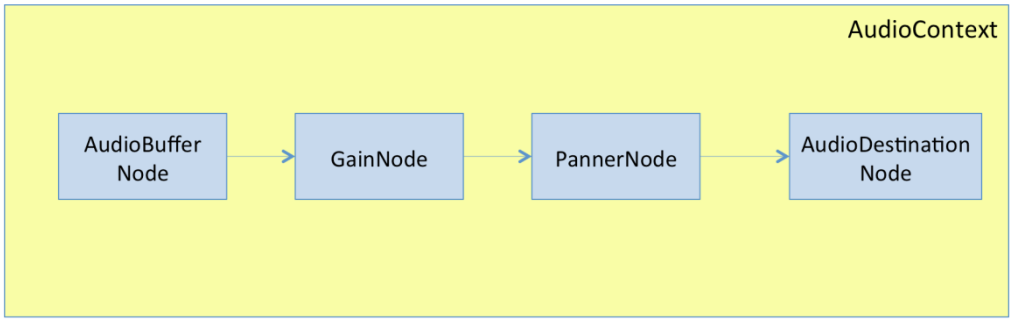



**Figure 9: A basic example of an audio processing graph using the native audio node objects provided by the WAA.**

The native audio *PannerNode* provides basic panning functions for stereo output as well as for binaural rendering over headphones with pre-encoded HRTFs. However, this binaural implementation presents several limitations. More efficient implementations have been proposed[5] and provide also means for loading personalised HRTFs encoded under the AES69 standard or HRTFs from publicly available database. More complex rendering algorithms over 3D loudspeakers layouts may require a specific audio rendering implementation as well. The WAA provides a means for writing custom audio effects directly in JS or WebAssembly. For a couple of years, such customised JS code would be handled by the *ScriptProcessorNode*, the major drawback of which was that it would run in the main UI thread, thus lacking efficiency (asynchronous, sensitive to UI interactions). Progressively, solutions are being developed to process the custom audio script within the audio rendering thread, using the *AudioWorklet* interface[6] (see Figure 10), thus providing better efficiency and accurate synchronisation between audio tasks.

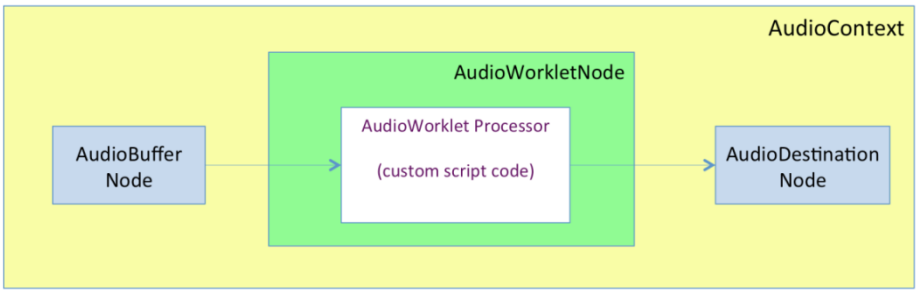



**Figure 10: Custom audio processing script code can be run within an AudioWorkletNode**

The main audio playback formats currently supported by the renderer are binaural over headphones and VBAP over conventional 2D or 3D loudspeakers layouts.

Even though browsers on mobile devices 'nowadays have many interfaces to the sensor APIs (such as velocity, e.g. for head-tracking) and GPS, it has to be taken into account that those are not always available on notebooks or desktop computers. Hence, the personalisation features need either to be dependent on the actual end device capabilities, or the user interface and the

[4] https://webaudio.github.io/web-audio-api/

[5] Carpentier, T., Binaural Synthesis with the Web Audio API, in proceedings of Web Audio Conference, Paris 2015

[6] https://developers.google.com/web/updates/2017/12/audio-worklet

personalisation features need to be the lowest common denominator, such as foreground/background balance.

Additional metadata for a particular programme user interface can be streamed in parallel to the AAC+ADM stream as additional JSON streams, possibly linking to resources outside of the object-based audio architecture specified in this document. For the sake of reducing redundant work, these additional JSON streams should be not tied to the specific NGA distribution codec capabilities, but rather be independent from them. In this way they could be used for any kind of object-based distribution (in the ORPHEUS case: AAC+ADM and MPEG-H).

# 4. For Further Study

Even though the ORPHEUS consortium tried to take into account as many real-world requirements and use cases as possible for the design and development of the reference architecture, some areas and scenarios could not, or could only partly, be explored during the project. The reasons for this are varied, and include missing standards, the lack of required infrastructure or limited capacities. However, the following paragraphs contain at least some guidance and expertise for those areas.

### *STORAGE: Archive systems*

A decisive component in broadcast workflows is a media asset management or archive system. For the migration to object-based audio it should support in any case ADM metadata and uncompressed audio formats. For audio-only archives, the BW64 container should be a reasonable choice, but depending on the product and workflow, other container formats such as RF64[7] would also fulfil these requirements. For audio-with-video archives, several choices are possible including MXF[8], IMF[9] or even BMF[10]. For MXF, however, Recommendation ITU-R BS.2076 contains some information about how to integrate ADM metadata. For BMF, the specification is currently being extended to support ADM.

### *RADIO STUDIO: Scheduling lists for consecutive programmes*

Typically, radio programmes are individually composed of a sequence of programme elements. These are planned in advance and played out according to a running order from a special component (hardware or software) in the radio studio. This component should support both file-based ADM and serialised ADM.

Several radio programmes are normally presented, according to a schedule, through the day. The play-out may be triggered by an automation system according to the schedule, or manually from a radio studio. The play-out of these entire programmes, when pre-recorded, requires similar capabilities for file-based ADM and serialised ADM, as well as advanced integrated metadata models (EBUCore) and adjacent descriptions (JSON as described in § 3.2.4).

### *DISTRIBUTION: Alternative formats and protocols*

Alternative format options exist in several places, but especially for the distribution of object-based content. For distribution to the end user, there is in particular HTTP Live Streaming (HLS) as an alternative to MPEG DASH. Considering Next Generation Audio (NGA) codecs, besides MPEG-H 3D Audio, there is also the Digital Audio Compression (AC-4) format, and DTS:X. In any case, all of those formats need to provide a suitable interface to ADM, potentially based on the "ADM Broadcast Emission Profile" introduced in § 3.2.5. The fact that these are not covered in the

[7] https://tech.ebu.ch/docs/tech/tech3306-2009.pdf
[8] http://ieeexplore.ieee.org/document/7292073/
[9] http://ieeexplore.ieee.org/document/7560857/
[10] http://bmf.irt.de/

ORPHEUS reference architecture in the same level of detail does not mean that they could not be used. Within the ORPHEUS consortium there were not the resources to investigate alternative formats in detail.

### *DISTRIBUTION: existing broadcast systems over non-IP platforms*

Even though the consumption of content on IP-based platforms is growing extensively, traditional broadcast emission like DAB (Digital Audio Broadcasting), DVB (Digital Video Broadcasting) or ATSC (Advanced Television Systems Committee) need to be taken into account for a holistic object-based workflow. Those emission channels are, and will continue to be, crucial for an unpredictable time period due to both technical and audience-acceptance reasons. At the present time, no activities within DAB were observable that one or more of the NGA technologies or codecs would be integrated in a new revision of the standard. For TV broadcast, however, both ATSC and DVB standards bodies published new revisions of their specification, which contain several NGA codecs[11,12] that may be implemented in TV devices. The broadcast infrastructure needs then to be updated with corresponding encoder and multiplexer devices. For terrestrial UHDTV services in Korea, ATSC 3.0 with MPEG-H 3D Audio, as the only NGA codec, has been standardized and was used during the Winter Olympics 2018 in PyeongChang. For this purpose, professional encoders from KaiMedia, PixTree and MediaExcel were developed, and TV sets from Samsung and LG with support of MPEG-H 3D Audio were available in stores.

### *DISTRIBUTION: Variable and various bitrates for distribution*

To reduce the average bandwidth required for distribution or emission, a variable bitrate is often used for the encoded audio. This feature was not used for the pilots, so no further details could be reported in the reference architecture. Nevertheless, the MPEG-H codec provides this functionality. It is clear that the use of audio objects that appear and then disappear might be a scenario where variable bit rates become important. Moreover, although MPEG-DASH was used, the pilots were encoded and streamed in one quality only: this could easily be done with MPEG-H, or other NGA codecs.

The implications of a reducing bitrate for individual audio channels are well known and didn't require detailed investigation.

### *DISTRIBUTION: additional content metadata*

Because a full and satisfactory object-based media user-experience exceeds the present capabilities of a purely object-based *audio* production and distribution format, additional metadata and matching presentation design is necessary (as described in § 3.2.4). The metadata that is required, or desirable, includes basic information about the programme, e.g. the schedule, title, description of the content, titles of elements included (music, artists, authors, contributors), as well as things such as visual elements (pictures) for illustration and navigation. Various approaches and solutions have been developed to enhance and enable legacy digital and analogue broadcast distribution systems. They offer similar user-experience features through integrated or hybrid (IP-connected) technologies.

Within the ORPHEUS project we've taken a 'design-oriented' approach, by identifying the requirements and features, regardless of what might already exist, in order to create and deliver the optimum for our pilots. Even so, it is likely that some of the basic features applied here could also be implemented with existing broadcast metadata technologies such as EPG[13], Dynamic Label

[11] https://www.atsc.org/standards/atsc-3-0-standards/

[12] http://www.etsi.org/deliver/etsi_ts/101100_101199/101154/02.03.01_60/ts_101154v020301p.pdf

[13] http://www.etsi.org/deliver/etsi_en/300700_300799/300707/01.02.01_60/en_300707v010201p.pdf

Plus[14] or RadioDNS[15]. Further investigation and evaluation is needed to identify their possible applicability and other practical solutions for future real-life broadcast implementation.

***REPRODUCTION: Object-based loudness***

The ORPHEUS partners conducted investigations to define an object-based loudness calculation and normalisation method but those have not been adopted by a standards organisation yet. Hence, no formal recommendation could be made for the reference architecture. It is foreseeable, however, that the ITU-R will publish a Recommendation in time, which will provide the necessary methodology. A loudness calculation and normalization of rendered loudspeaker feeds according to ITU-R BS.1770-4[16] was used for the pilots, which provided an appropriate workaround.

## 5. Conclusions

Broadcasting architectures and infrastructures have been developed, refined and optimised over the last 100 years. Up to now, the final product delivered to the audience, was assembled edited, mixed and configured as an immutable entity, leaving few or no options for adjustments according to the reception conditions and the personal preferences of the recipient.

The object-based media approach fundamentally changes this: A production is now a collection of media assets, so-called ‘objects’, along with ‘metadata’ describing their attributes and relationships. This package is delivered to an end user device, where it gets optimally ‘rendered’ according to the capability of the device, environmental conditions and user preferences.

It seems obvious that this alternative concept will cause profound changes for existing broadcast system architectures and infrastructures, as well as their related workflows. Yet, it was not the task for the ORPHEUS project to create such a new system from scratch.

Therefore, in order to create a universal Reference Architecture for a realistic end-to-end object-based audio broadcast chain, the ORPHEUS consortium took the well-established components and efficient workflows of the existing broadcasting eco-system as initial basis.

By this, we have incorporated a plethora of requirements from real-world broadcast use cases and from multiple pilots. These have been integrated and implemented successfully into our new object-based approach. As demonstrated in the pilots, several different scenarios have been realised, enhancing traditional radio broadcast formats with new object-based audio features and thus creating exiting new user experiences.

The outcome of these pilots was the basis for the ‘practicable’ Reference Architecture, designed and developed in several iterations, based on lessons learned from the various implementations during the project.

The Reference Architecture presented here can be taken as a genuine template for broadcasters and media creators for starting their transition process into object-based audio production and distribution workflows.

## 6. Acknowledgment

This work was funded in part from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 687645 (ORPHEUS project). The paper reflects only the authors' views. The Commission is not responsible for any use that may be made of the information

[14] http://www.etsi.org/deliver/etsi_ts/102900_102999/102980/01.01.01_60/ts_102980v010101p.pdf

[15] http://www.etsi.org/deliver/etsi_ts/103200_103299/103270/01.02.01_60/ts_103270v010201p.pdf

[16] https://www.itu.int/rec/R-REC-BS.1770-4-201510-I/en

therein.

# 7. References


[1] ORPHEUS project Deliverable D2.4, “Final Reference Architecture Specification and Integration Report”, 2018: https://zenodo.org/record/1204867

[2] Scheirer E., Vääänänen R. 1999. AudioBIFS: Describing Audio Scenes with the MPEG-4 Multimedia Standard, in IEEE Transactions On Multimedia 1, Nr. 3

[3] Geier M., Spors, S. 2008. ASDF: Audio Scene Description Format. International Computer Music Conference (ICMC), Belfast, UK

[4] Peters N. et al. 2012. SpatDIF: Principles, Specification, and Examples. Proceedings of SMC 2012

[5] Peter Brightwell, Jonathan Rosser, Robert Wadge, Phil Tudor, "The IP Studio", BBC R&D White paper WHP 268, Sept. 2013

[6] http://www.bbc.co.uk/rd/projects/ip-studio

[7] ITU-R Recommendation BS.2076, “Audio Definition Model”, https://www.itu.int/rec/R-REC-BS.2076/en

[8] ITU-R Recommendation BS.2094, “Common Definitions for the Audio Definition Model”, https://www.itu.int/rec/R-REC-BS.2094/en

[9] ORPHEUS project Deliverable D4.2, “”, 2017: https://zenodo.org/record/845336

[10] Bleidt, R.L., D. Sen, A. Niedermeier, et al., in “Development of the MPEG-H TV Audio System for ATSC 3.0.” IEEE Transactions on Broadcasting, 2017. 63(1): p.202..236, DOI: https://doi.org/10.1109/TBC.2017.2661258.

[11] ORPHEUS project Deliverable D2.3 “Interim Pilot Progress Report”, 2017, https://zenodo.org/record/844003

# Annex A: Supplementary information

The following information has been compiled from various sources that are noted in each section.

## *A1 Channel-based & Object-based audio:*

*(Source: https://lab.irt.de/demos/object-based-audio/index.html)*

In channel-based audio (Figure A1), each audio channel in the final audio programme has to be reproduced by a loudspeaker at a well-defined position. This fixed audio mix is transmitted to the end-user who has no real means to adapt its replay to his/her needs or personal preferences. The replay device may also be dictated by the programme (stereo, 5.1 etc.)

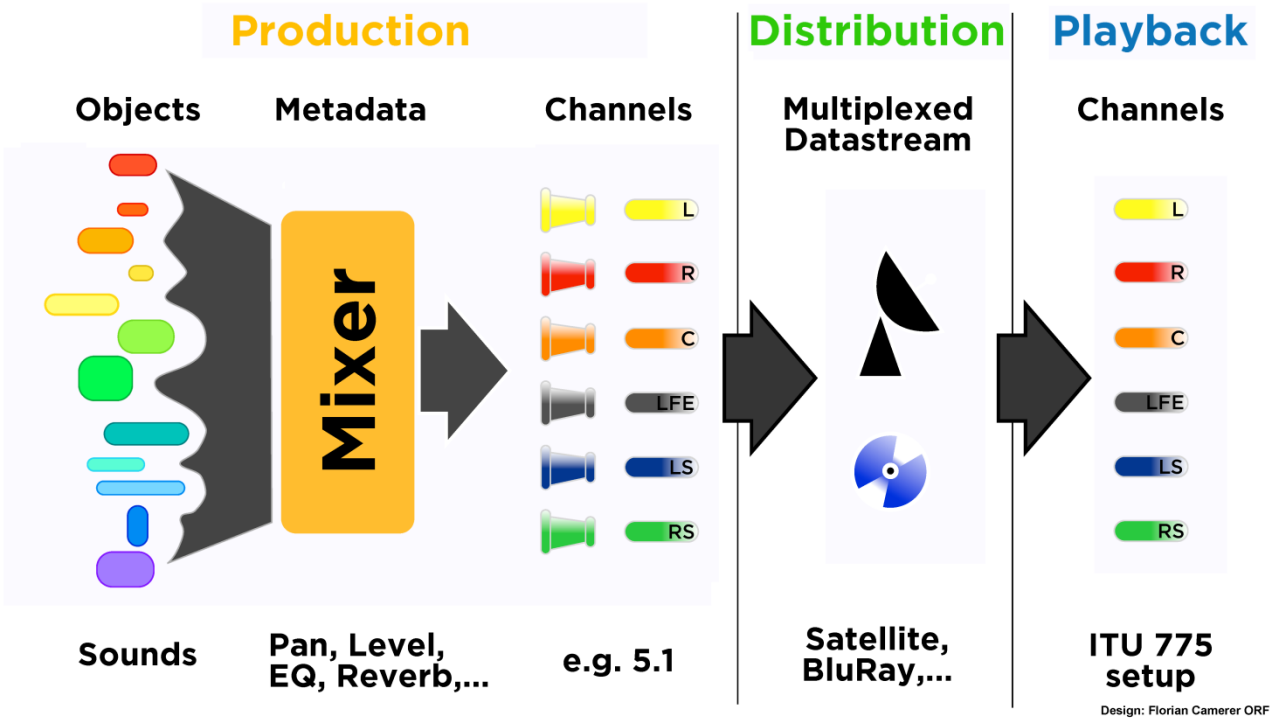



**Figure A1: Principle of channel-based audio**

An object-based production approach is able to overcome the abovementioned obstacles. The term 'object-based media' has become commonly used to describe the representation of media content by ‘a set of individual assets together with metadata describing their relationships and associations’ (i.e. the objects). At the point of consumption these objects can be assembled to create an overall user experience. The precise combination of objects can be flexible and responsive to user, environmental and platform-specific factors. For playback, the object-based content needs to be 'rendered' to the reproduction layout, such as a multi-channel loudspeaker set-up. The general principle of object-based audio is illustrated in Figure A2.

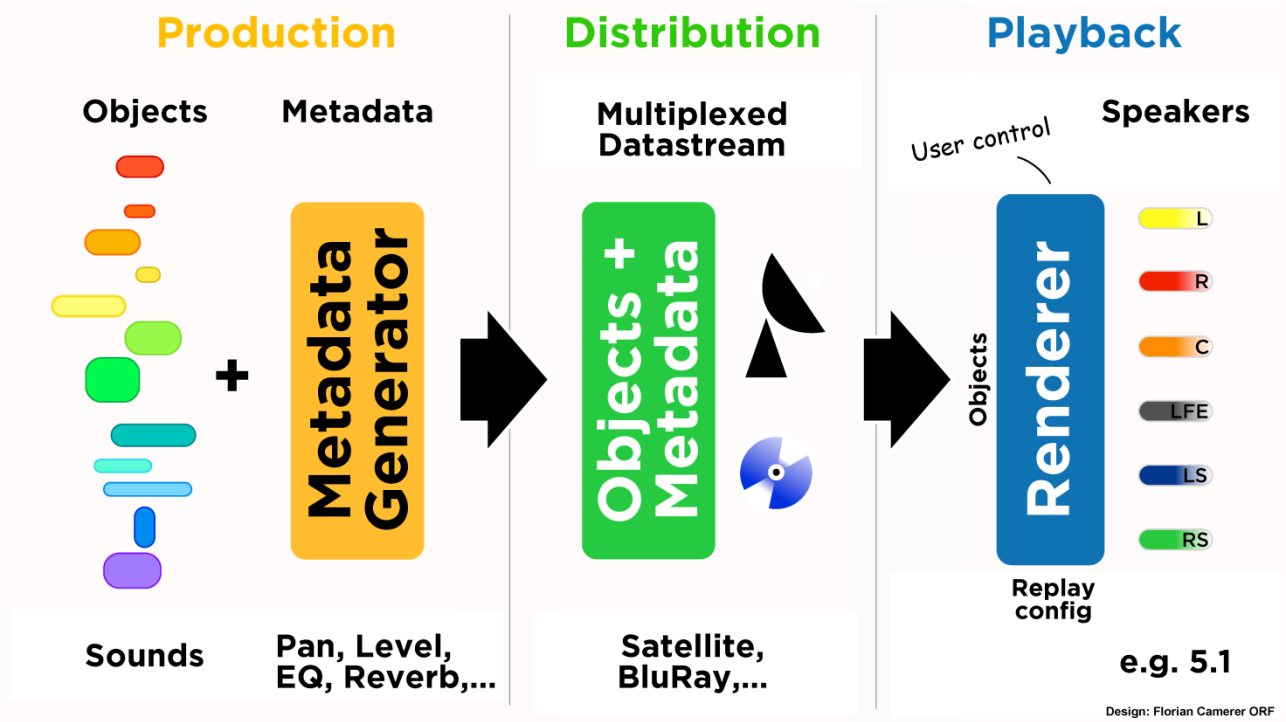



**Figure *A*2: Principle of object-based audio**

The term 'rendering' describes the process of generating actual loudspeaker signals from the object-based audio scene. This processing takes into account the target positions of the audio objects, as well as the positions of the speakers in the reproduction room. It may further take into account user interaction such as a change of position or level. An object-based approach can therefore serve end-users more effectively, by optimizing the experience to best suit their access requirements, the characteristics of their playback platform and the playback environment or their personal preferences.

More than this, it will be highly beneficial for content producers, as workflows can be streamlined and only a single product needs to be created, archived and transmitted in order to support and serve a multitude of potential target devices and environments. This is enabled by the simple fact that the metadata of individual objects can be modified and adjusted, either by the end-user or along the production and transmission chain, without the need to change the audio material itself. This way, the four key features of object-based media – interactivity and personalization, accessibility, immersive experiences and compatibility – can be achieved in a non-destructive, controlled and scalable way.

Current Next Generation Audio (NGA) distribution systems and production formats support also combinations of channel-based and object-based elements. This so-called hybrid approach allows the content producer to create a channel-based “bed” and enrich it with separate objects, see Figure A3.

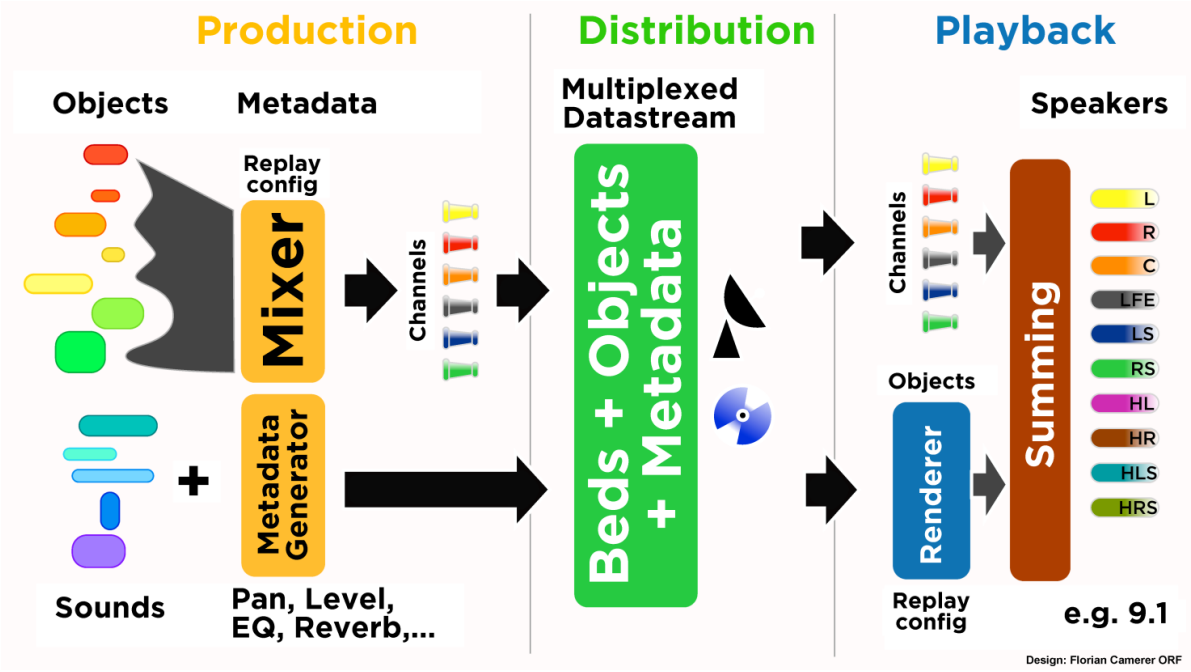



**Figure A3: Hybrid approach using channel-based audio along with object-based audio**

***In summary:***

Object-based audio is a revolutionary approach for the creation and deployment of interactive, scalable, immersive and cross media applications for any type of media content. This is possible by representing the audio content as a set of individual assets together with metadata describing their relationships and associations. The most important features are:

- Personalization of content representation (e.g. speech level customization or multi-lingual features)
- Immersive experience for any kind of content
- The delivery of audio content in a format- and system-agnostic manner.

## *A2 Recommendation ITU-R BS.2076 'Audio Definition Model'*

The Audio Definition Model is an ITU metadata specification that can be used to describe object-based audio, scene-based audio and channel-based audio. It can be included in BWF-compatible audio files or used as a streaming format in production environments.

## *A3 DVB & DVB TS 101 154*

DVB is an industry-led consortium of the world's leading digital TV and technology companies, such as manufacturers, software developers, network operators, broadcasters and regulators, committed to designing open technical specifications for the delivery of digital TV.

TS 101 154 is the Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream; it provides implementation guidelines for the use of audio-visual coding in satellite, cable and terrestrial broadcasting distribution systems that utilize MPEG-2 Systems. SDTV, HDTV, UHDTV using HEVC coding, Frame Compatible Plano-Stereoscopic 3DTV and Full Resolution HD 3DTV using MVC Stereo are covered.

The specification covers the first and second phases of the DVB UHDTV specification, as well as DVB Next Generation Audio specification. The most recent version also contains H.264/AVC and HEVC

video conformance points for use with MPEG-DASH. These are aligned with the broadcast conformance points, supporting a similar feature set, but take into account the specific requirements of adaptive bitrate delivery over IP-based networks.

For NGA, TS 101 154 has included both AC-4 Part 2 and MPEG-H; DTS will be added shortly.

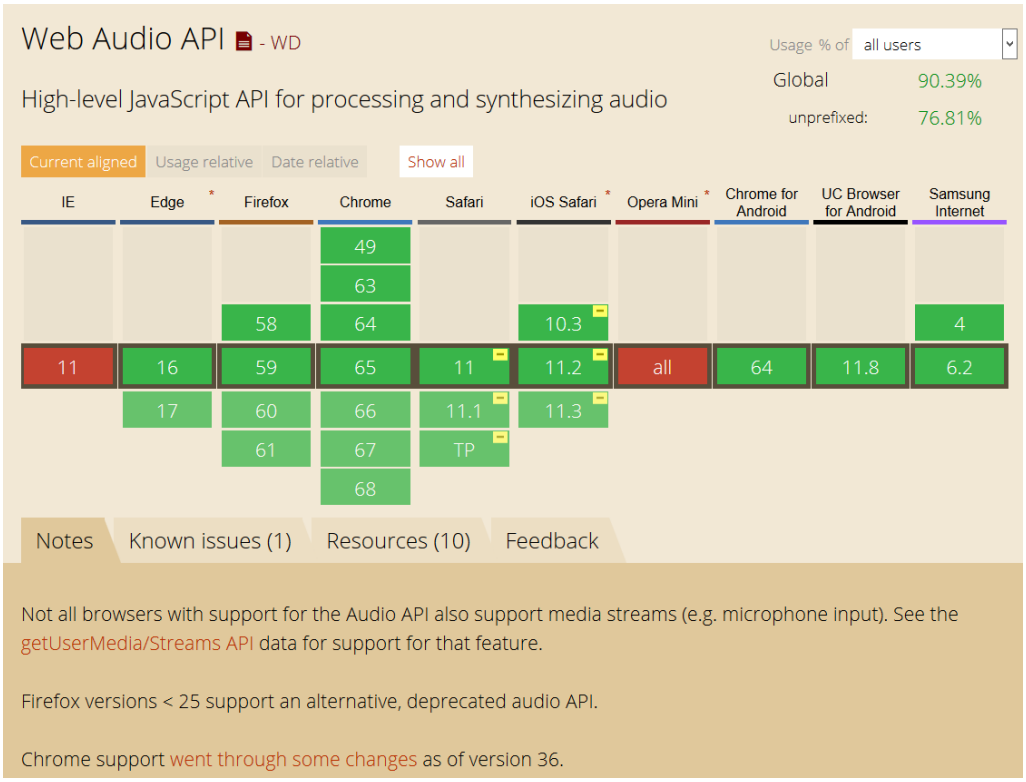
## *A4 HTML5 & WebAudio API*

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current major version of the HTML standard. It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc.

HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications because it includes features designed with low-powered devices in mind. Many new syntactic features are included.

To natively include and handle multimedia and graphical content, the new <video>, <audio> and <canvas> elements were added, and support for scalable vector graphics (SVG) content and MathML for mathematical formulas. To enrich the semantic content of documents, new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav> and <figure>, are added. New attributes are introduced, some elements and attributes have been removed, and others such as <a>, <cite> and <menu> have been changed, redefined or standardized.

Web Audio API is a high-level JavaScript API for processing and synthesizing audio. The figure below depicts the browsers supporting the Web Audio API.




Source: https://caniuse.com/#feat=audio-api

## *A5 MPEG-DASH*

Source https://www.encoding.com/mpeg-dash/

Adaptive bitrate streaming has become the standard for delivering video content online to multiple devices. This type of delivery is a combination of server and client software that detects a client's bandwidth capacity and adjusts the quality of the video stream between multiple bitrates and/or resolutions. The adaptive bitrate video experience is superior to delivering a static video file at a single bitrate, because the video stream can be switched midstream to be as good or bad as the client's available network speed (as opposed to the buffering or interruption in playback that can happen when client's network speed can't support the quality of video). Because it uses the standard HTTP port, the lack of firewalls, special proxies or caches, and its cost efficiency have increased its popularity and use.

There are three main protocols for this type of delivery- HTTP Live Streaming, Microsoft Smooth Streaming, and HTTP Dynamic Streaming. Each protocol uses different methods and formats, and therefore, to receive the content from each server, a device must support each protocol. A standard for HTTP streaming of multimedia content would allow a standard-based client to stream content from any standard-based server, thereby enabling consistent playback and unification of servers and clients of different vendors.

In response to the scattered landscape, MPEG issued a Call for Proposal for an HTTP streaming standard in April 2009. In the two years that followed, MPEG developed the specification with participation from many experts and with collaboration from other standard groups, such as the Third Generation Partnership Project (3GPP). More than 50 companies were involved — Microsoft, Netflix, and Adobe included — and the effort was coordinated with other industry organizations such as studio-backed digital locker initiator Digital Entertainment Content Ecosystem, LLC (DECE), OIPF, and World Wide Web Consortium (W3C). This resulted in the MPEG-DASH standard being developed.