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

ORPHEUS
Object-Based Audio Experience

Object-based broadcasting – for European leadership in next generation audio experiences

D2.2: Interim Reference Architecture Specification and Integration Report

Version: v2.0

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	30/04/2017
Submission date	05/05/2017
Lead editor	Michael Weitnauer (IRT)
Authors	Michael Weitnauer (IRT), Chris Baume (BBC), Andreas Silzle (FHG), Nikolaus Färber (FHG), Olivier Warusfel (IRCAM), Nicolas Epain (BCOM), Tilman Herberger (MAGIX), Nikolaus Färber (FHG), Benjamin Duval (TRI), Niels Bogaards (ECANDY)
Reviewers	Halid Hrasnica (EURES), Werner Bleisteiner (BR)
Work package, Task	WP2, T2.1, T2.2
Keywords	Architecture, workflow, specification

Abstract

Deliverable D2.2 provides an overview about the current state of the pilot implementation architecture and its influence on the final reference architecture. Moreover, the integration activities so far are summarised. The reference architecture will be developed during the project time, based on experiences from the project pilots. A detailed explanation regarding the distinction between reference architecture and pilots is included. The planned workflow of the pilots is described as well as the current state of macroblocks and their components. This version of D2.2 includes updates, which were requested by the EC reviewer after the first submission. Hence, interfaces were identified and where necessary, described in more detail. This document will be updated once more during the project.

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1	15/12/2016	Initial version	IRT
v0.3	18/01/2017	Merged contributions from BBC, BCOM, IRCAM and FHG	
v0.4	19/01/2017	Additional input for section 3	BBC
v0.5	19/01/2017	Discussion results implemented and changes so far accepted	
v0.6	24/01/2017	First round of review comments addressed & additional small updates of certain sections	BR, IRT, BBC
v0.7	25/01/2017	Second round of review comments addressed	IRT
v0.8	25/01/2017	Section 3.2 and 3.3 updated – version for PMC approval	
v1.0	30/01/2017	PMC approved – ready for submission	
v1.1	07/03/2017	1st Review feedback: modifications prepared	IRT
v1.2	03/04/2017	Contributions form BBC, BCOM and MAGIX included	BBC, BCOM, MAGIX, IRT
v1.3	10/04/2017	Integration of meeting decisions	IRT
v1.4	18/04/2017	Contributions for macroblock subsections	BBC, BCOM, MAGIX, FHG, IRT, TRI, ECANDY
v1.5	19/04/2017	Sections 2.2.1 and 2.2.5 updated	BCOM, IRCAM
v1.6	20/04/2017	Diagrams updated	BBC, IRCAM
v1.7	27/04/2017	Internal review addressed	
v1.8	02/05/2017	Some typo corrections	IRCAM
v2.0	05/05/2017	Final editing and submission	EURES

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

One of the major objectives of ORPHEUS is the specification of a reference architecture for an end-to-end object-based production workflow. The intention of such a reference architecture is to provide a guideline for other content producers and broadcasters for building an object-based production and distribution workflow and how to adapt established infrastructure.

To achieve this, the consortium is currently in the process of specifying the pilot implementation architecture that will be used for the pilots in the project. Based upon findings and lessons learned from ORPHEUS pilots, the reference architecture will be specified later during the project. It will be – compared to the pilot implementation architecture – rather format and interface agnostic wherever possible and reasonable in order to be applicable as a general guideline for other broadcasters.

This Deliverable is an update of D2.1 which reported the Architecture Specification activities until M7. Compared to D2.1, this document contains several updates in subsections 2.3 – 2.7 as well as a new section which describes the integration activities for the pilot phase 1 architecture until M14.

The already planned blocks, components and interfaces of the pilot implementation architecture are described along. Moreover, the current state and decisions taken so far are included. Eventually, the integration activities until M14 for running the first pilot are reported in this document.

This document also provides updates which were requested by the reviewers of the European Commission after the first year project review. To satisfy these requests, more details on interfaces, formats and protocols were included.

Table of Contents

List of Figures

Abbreviations

DAW	Digital Audio Workstation
PCM	Pulse Code Modulation
ADM	Audio Definition Model
VST	Virtual Studio Technology
HOA	Higher Order Ambisonics
BW64	Broadcast Wave 64 Bit
IP	Internet Protocol
DAB+	Digital Audio Broadcasting +
DVB-S	Digital Video Broadcasting Satellite
ITU	International Telecommunication Union
MPEG	Moving Pictures Expert Group
AAC	Advanced Audio Coding
DASH	Dynamic Adaptive Streaming over HTTP
MP4	MPEG-4 File Format
DASH-IF	DASH Industry Forum
UI	User Interface
NMOS	Networked Media Open Standards
AES67	Audio Engineering Society standard for audio over IP
UMCP	Universal Media Composition Protocol
JSON	JavaScript Object Notation
RTP	Real-Time Transport Protocol
GPS	Global Positioning System
VBAP	Vector Base Amplitude Panning
EBU	European Broadcasting Union
API	Application Programming Interface
gRPC	Google remote procedure call
NGA	Next Generation Audio
MPD	Media Presentation Description
MSE	Media Source Extension
CSS	Cascading Style Sheets
ATSC	Advanced Television Systems Committee
DRC	Dynamic Range Control
IIR	Infinite Impulse Response Filter
FIR	Finite Impulse Response Filter

1 Introduction

This Deliverable is an update of D2.1 which reported the Architecture Specification activities until M7. Compared to D2.1, this document contains several updates in subsections 2.2.1, 2.2.2, 2.2.3, 2.2.4 and 2.2.5 as well as a new section which describes the integration activities of the ORPHEUS partners for the pilot phase 1 architecture until M14. Moreover, a short outline of planned internal integration milestones is enclosed in section 3.2.

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 recording and broadcasting audio. 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][2][3]) have utilised some variation or parts of the object-based concept. This is due to the fact that previous attempts did not utilize the full creative and technical process from planning, editing, production and consumption of the object-based audio. This has led to isolated, partial solutions that did not take into account the whole or 'big picture' as required to unlock the full potential of an object-based production approach.

The ORPHEUS project therefore opts for an integrated method, targeting 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 developed concepts 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.

The core of this concept is formed by a pilot and a reference architecture for object-based media production. Thus, the reason for having a real-world pilot in the centre of the concept is not only for demonstration and evaluation purposes. Furthermore, it defines the requirements for an end-to-end chain that utilises the innovative features of object-based media while enforcing the constraints of a real-world, day-to-day production workflow. This approach ensures that every step of the chain will be considered with all shareholders in discussion to make the appropriate development possible.

The envisioned pilot will be implemented using the reference architecture developed and specified within the project. This reference architecture will provide specifications for the interfaces and components required for an end-to-end production chain. This will be beneficial beyond the projects lifetime, as it can serve as a guideline to set up, integrate or migrate an object-based production workflow. Apart from that, it can be used as a benchmark or test bed, ensuring the interoperability of potential future improvements of individual components within the chain. Work on the reference architecture and the implementation of the pilot will lead to an in-depth understanding of an object-based workflow and provide the tools required for creative object-based production.

1.2 Role of T2.1 for the project

WP2 can be seen as the 'umbrella' work package of ORPHEUS, as it defines interfaces, receives feedback from other work packages and makes decisions for the specific pilot implementations, which are driven by the use cases. Figure 1 illustrates the central role of WP2 and T2.1 for the entire project.

The aim of Task 2.1 is the definition of interfaces, requirements and specifications of a reference architecture for an end-to-end object-based audio broadcasting chain. Certain tools of the implementation will be made available on an open-source basis if possible. The reference architecture will be a guide for future implementations of tools and building blocks, taking into

account defined interfaces and best practices, as well as potential pitfalls and newly created standards and working practices by the project partners. Furthermore, it can be used as a reference point to test and evaluate new developments. An initial version of this reference architecture will be the basis for both phases of the pilot and will be refined further, based on the results of the integration and testing process. It will also give the clearest indication yet of how media organisations can adapt to operate this technology at the scale required to run broadcast stations/services 24 hours a day, 7 days a week, 365 days a year. Although broadcasters are very experienced in this scale of operation, these challenges are not trivial. The technology will fundamentally change monitoring, automation, workflows and asset management and few organisations have the knowledge required to manage the metadata from capture to consumption.

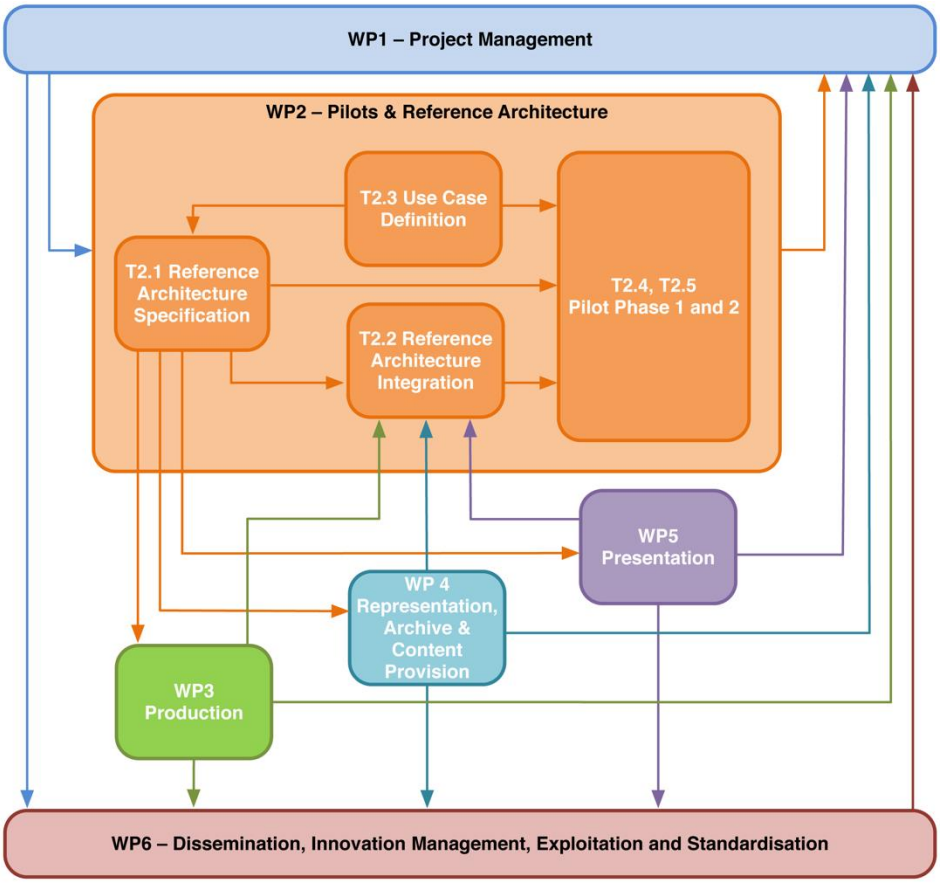

Figure 1: Pert chart of overall ORPHEUS WP structure

2 Architecture Specification

2.1 Relation between pilots and reference architecture

One major goal of WP2 is the definition and specification of a reference architecture for object-based audio broadcasting. In order to avoid confusion, it has to be distinguished between the reference architecture and the ORPHEUS specific pilot implementation architecture.

The reference architecture will provide specifications of the required interfaces and components for an end-to-end production chain and will be beneficial beyond the projects lifetime, since it can serve as a general guideline to setup, integrate or migrate an object-based production workflow. Further, the reference architecture will be format, interface and protocol agnostic as much as possible and reasonable to be useful for a large number of broadcasters as their infrastructure can be very different. We will rather, wherever possible, list different options for formats, tools or components of the reference architecture. On the other hand, the specific pilot implementation architecture will be based on the established infrastructure of the broadcaster partners in the project, on the available tools of other consortium members and on further technical outcomes of the project.

Therefore, the reference architecture will be based as much as possible on existing channel-based workflows since these workflows are the result of long-term experiences of broadcasters and content producers that guarantee the best quality of the final product in an economic manner.

The focus of WP2 and T2.1 lied on the specification of the pilot implementation architecture with the implications on the reference architecture in mind. Further, within ORPHEUS it will be the first time that a broadcasting workflow will be assembled for a complete end-to-end object-based structure and no experiences are available yet. Hence, for the definition and specification of a reference architecture, the actual pilot implementations are essential to gather the knowledge and experience for the universal, implementation-agnostic reference architecture. This will ultimately lead to a step-by-step development of the final reference architecture at the end of the project.

2.2 Planned components and signal flow of the pilot implementation architecture

Figure 3 illustrates the current state of the planned pilot implementation architecture, with a signal flow graph. The graph contains the five identified so-called macroblocks:

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

These blocks are described in more detail in the subsections below. The macroblocks are independent from each other, have several smaller components and are somewhat coherent with the ORPHEUS work package structure. Even though the main venue will be at the BBC premises in London, each macroblock will potentially be situated at the venue of a different project partner. This is especially true for the Pre-Production and the Reception blocks. Interfaces and interconnections between macroblocks and their components are also illustrated in Figure 2.

Compared to D2.1, the sections 2.3 – 2.7 have been updated with respect to new agreements, investigations or achievements. Since some parts of the planned pilot architecture components or tools were not yet entirely clear in M7, the ORPHEUS partners worked on these gaps since then and provided the results as updates in this document.

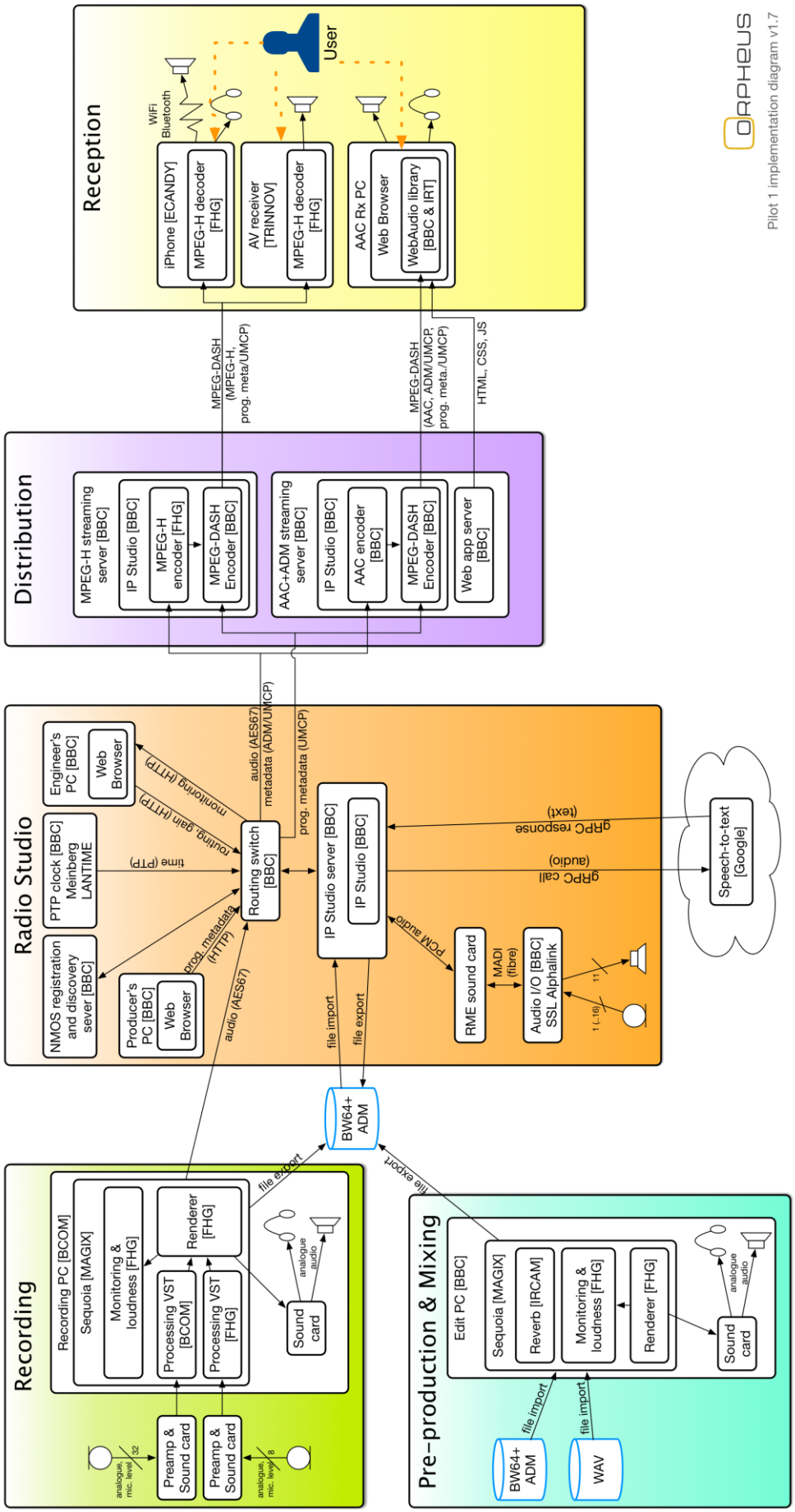

Figure 2: Current status of pilot implementation workflow overview

2.2.1 Recording

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

- Strictly speaking object-based audio, i.e. mono or stereo tracks along with metadata describing the position, width, etc. of the object.
- 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 speaker positions. This includes stereo but also more spatialized formats like 7.1+4.

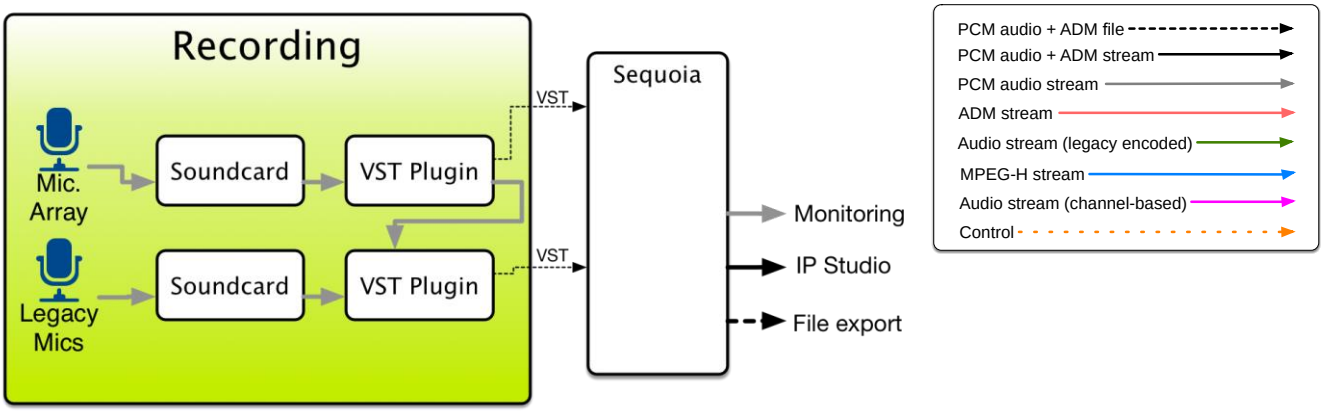

Figure 3: Example of recording block structure for Phase 1 of the Pilot.

In the case of non-live productions, the audio data produced within the recording macroblock may be stored for later use. In this case, the audio is stored as WAV or BW64 files with ADM metadata. However, for live productions or when the producers wish to mix the audio signals prior to recording, the recording block shares an interface with the pre-production and mixing block. In this case, the recording block sends audio data *directly* to the DAW, which is used to monitor and/or mix the recorded signals, and possibly add or edit the corresponding metadata. This is how the two blocks will communicate with each other for Phase 1 of the Pilot, as it is a live production. An example of structure for the recording block is illustrated in Figure 3.

The envisioned scenario for phase 1 of the pilot is to mix a number of mono/stereo objects, which will be recorded using traditional proximity microphones, with a channel-based or scene-based bed recorded with a microphone array. The objects will consist of the sources of interest in the scene (typically someone speaking) while the multi-channel bed will constitute the ambient part of the scene. For the mono and stereo sound objects, the recorded signals will be sent directly from the sound card used to digitize the signals to the DAW, using the sound card driver (typically an ASIO driver). For the multichannel bed, however, signals recorded using the microphone array have to be processed to produce scene-based or channel-based content. This will be done using the plugins developed by FhG and b<>com during the course of the ORPHEUS project. This plugins interface with the DAW via Steinberg's **VST** standard.

Below is a list of the VST plugins that may be used for Phase 1 of the Pilot:

- b<>com's MicProcessor, a plugin that converts signals recorded by microphone arrays to HOA signals.
- b<>com's Render Hoa2Spk, which renders HOA signals to speaker signals. This plugin would be used to produce a channel-based bed.
- b<>com's MainSpot, a plugin that estimates the position (direction and distance) of a

proximity (spot) microphone relative to a microphone array (main microphone). This is to help producing object metadata.

- b<>com's HoaScope, a monitoring plugin which analyses recorded HOA signals and displays a map of the acoustic energy as a function of the direction of arrival.
- FhG's 3D audio capturing plugin, which processes signals recorded by FhG's compact circular microphone array to speaker signals (channel-based format). This plugin would be used to produce a channel-based bed.

In the example illustrated in Figure 3, the signals recorded by the microphone array are processed by the MicProcessor plugin to obtain HOA signals. These signals are transmitted to Sequoia for mixing, as well as to the MainSpot plugin where they are analysed together with the signal recorded by a proximity microphone.

Interfaces to other components:

The Capturing macroblock has the following interfaces to other macroblocks:

DAW: If the plugins are used by the production teams, the raw signals from the microphone array plugins are converted to HOA or channel based audio using VST-2 plugins in a 32-channel bus. Other mono and stereo microphone signals are imported in the DAW via a sound card or as WAV or BW64 files.

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

2.2.2 Pre-production and Mixing

The purpose of the pre-production and mixing block 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 block is the object-based DAW, which is extended by several tools and workflows to import, edit, monitor and export object-based content. These tools and workflows are developed and implemented in task T3.2. The DAW will need to interact with the components presented in the section below and illustrated in Figure 4.

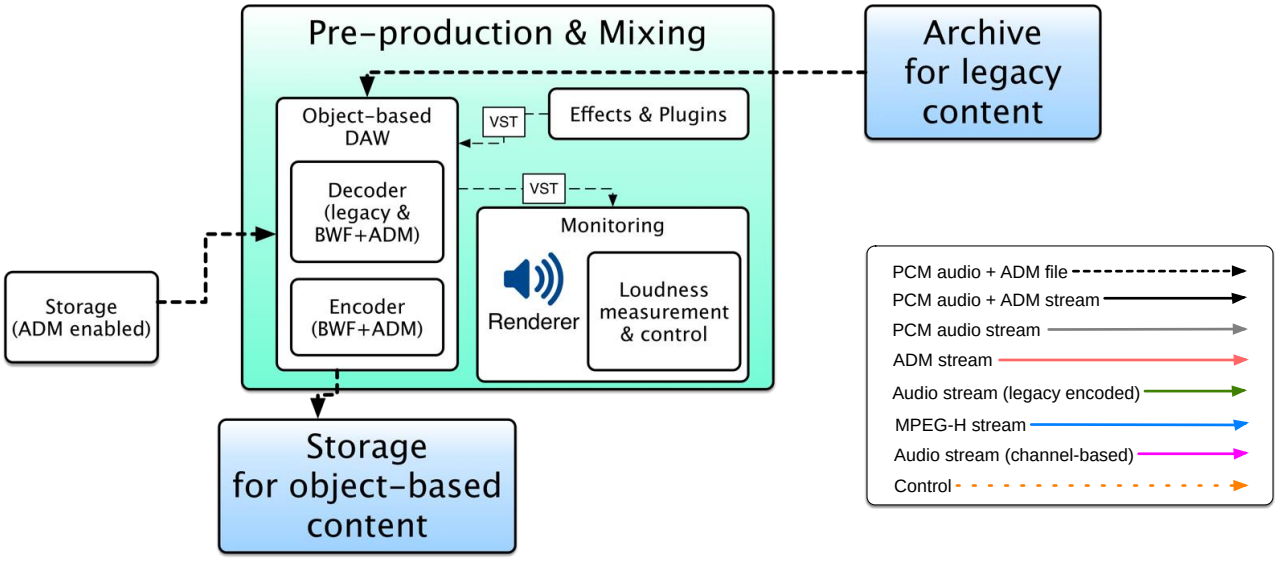

Figure 4: Current status of Pre-production & Mixing macroblock

Description of DAW components:

The following main components of the object based DAW will be configured for the pilot 1:

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 may be stored in a temporary **ADM enabled storage** or directly in the **storage for object-based content.**

Additionally, there will be a way to record object-based content directly within the DAW. This implicates further requirements for the DAW (e.g. 32 input channels for the microphone array processing described in the recording macro block).

Storage for object-based content

The DAW will 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. There will be tools to edit existing ADM metadata or additional metadata required for broadcasting.

Archive for legacy content

For integration into an established broadcasting architecture the access to existing content is very important. For this purpose, there have to be tools inside or outside the DAW to convert legacy content to object-based content by adding the required metadata to it. This process should at least partially be done automated - if possible. Even though some options were discussed, it is still to be investigated how an automated process should work.

Effects and plugins

A typical application for a DAW is the appliance of effects or plugins onto audio tracks. Basically, applying an effect consists in taking a given sound signal and changing it somehow. Typical effects used in audio production are filtering effects (changing the frequency content of a sound) or dynamic effects (limiter or compressor, to change the dynamic of an audio track). Plugins are extensions for the DAW to be used with audio tracks such as special filter effects.

Usually, effects and plugins can be applied to audio content within the DAW. Due to the nature of the ADM format, there are some limitations to consider here. Especially there is no final mastering stage for a certain channel format, but effects can be applied to individual audio objects only.

For the scope of ORPHEUS, it is decided that any relevant plugins should be made available as VST 2 plugins – at least for the pilots. This may also include format conversions e.g. converting 3D microphone input to HOA format or converting HOA format to channel-based surround formats. Plugins for these use cases already exist and can be delivered e.g. by B-COM.

A special use case is applying reverb to the content. Currently reverb 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. An object-based reverb algorithm and inclusion into ADM format is investigated by IRCAM in the scope of T3.2 and first results have already been published in D3.2.

To deal with 3D spatial audio the panning engine of the DAW has to be enhanced to handle 3D automation curves and convert them from and to the ADM objects as well as to the renderer plugin.

Monitoring

For previewing the object-based content and simulating the user experience it is important to allow monitoring with different speaker setups (including binaural monitoring). For this purpose, an ADM renderer needs to be implemented in the DAW.

As the ITU-R WP6C is currently in the process of standardizing a baseline renderer for the ADM interpretation, this option should be taken for the reference architecture. However, it will not be available before mid-2017. Meanwhile, it is decided to use the MPEG-H renderer solution of FHG, especially for the development of the pilots.

An essential part of the monitoring is **loudness measurement and control.** This may be either supported by additional VST plugins in the DAW and is researched in T3.4 or implemented by using the built-in tools of the MPEG-H production lib, which is integrated in Sequoia.

In addition to pure audio monitoring, other parts of the user experience may be simulated within the DAW, e.g. transport control or exchanging language specific objects. This will be researched in T3.4 and possibly in T5.4.

Another solution for the monitoring is to stream the content as an ADM compatible stream (as dealt with in T4.2), e.g. to support alternative real-time monitoring renderers or integration into live production environments (T3.3/3.4).

Use cases for Pilot 1:

For the first pilot the following DAW related use cases are going to be implemented:

ADM Create / Export / Import: BW64 files including ADM metadata can be created from scratch, imported to DAW internal data structures and exported from the internal format. The user interface for these tasks uses track naming conventions.

This feature is used for the pilot to pre-produce content and to exchange object based content with the partners. Also legacy content can be imported and converted into object based content by adding metadata, e.g. interactivity features and 3D positioning data.

3D Audio Playback and Rendering via MPEG-H renderer: The DAW will be enhanced to handle 3D object based content by adding a 3D panning module and enhanced automation curve handling.

All needed audio and metadata is passed from each audio object to a multichannel renderer plugin (MPEG-H renderer) which creates a standardized output signal in various formats, e.g. 7.1+4, 5.1, 2.0 and optionally binaural. This ensures that the audio producer can easily preview the resulting audio in various formats.

Realtime streaming: In the second phase of pilot 1, audio and metadata will be streamed to the Broadcast Studio software using IP protocol AES 67 and UMCP. Using these protocols, the DAW can act as a real-time player software for the live broadcast studio use case.

Interactivity features:

The following interactivity features of the resulting end user audio stream will be supported by the DAW:

- **Language tagging**: multiple dialog languages can be part of one project, which can be selected exclusively and are added to common tracks, e.g. a music or background bed. Language tagging will be done via special track comments.

- **Foreground / background tagging**: In the DAW it will be possible to tag several objects as foreground and other objects as background. This gives the end user the option of balancing between these objects, e.g. enhance the volume of a speaker and reduce the volume of a background sound within a range of producer assigned limitations. Foreground / background tagging will be done via object comments. Un-tagged objects are not affected by the balancing operation.

- **Chapter marker**: In the DAW it will be possible to mark the beginning of new chapters in one project using chapter markers. This gives the end user the possibility to rewind to the beginning of each previous chapter or skip forward to the next chapter – if not in a live broadcast. Chapter markers can be set graphically on the marker timeline in the DAW.

Interfaces to other components:

The DAW macroblock has the following interfaces to other macroblocks:

File Archive: BW64 + ADM files can be exported to and imported from the file archive using network file system access.

Legacy Content Archive: Existing Wav, MP3 or other audio files can be imported using file system and be converted into BW64 + ADM.

B-COM microphone array signal conversion: The raw signals from the 3D microphone array are converted to HOA or channel based audio using VST-2 Plugins in a 32 channel bus.

Renderer integration: The MPEG-H renderer plugin is integrated as an internal VST-2 plugin in the DAW with multiple inputs and one multi-channel output. In each track a metadata reader is installed and the audio renderer plugin is installed in a multi-channel surround output bus.

Radio Studio: For a live connection between the DAW and the IP based broadcast studio in the 2nd phase of pilot 1 the AES-67 protocol is used to transfer the audio and the UMCP protocol to simultaneously transfer the related metadata.

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

2.2.3 Radio Studio

There are three main aims to the radio studio macroblock: capture and control, monitoring and record/playback. Each of these is explained below, with detail about the functionality within each. We then describe the interfaces used to communicate between components and finish by giving a quick overview of our implementation.

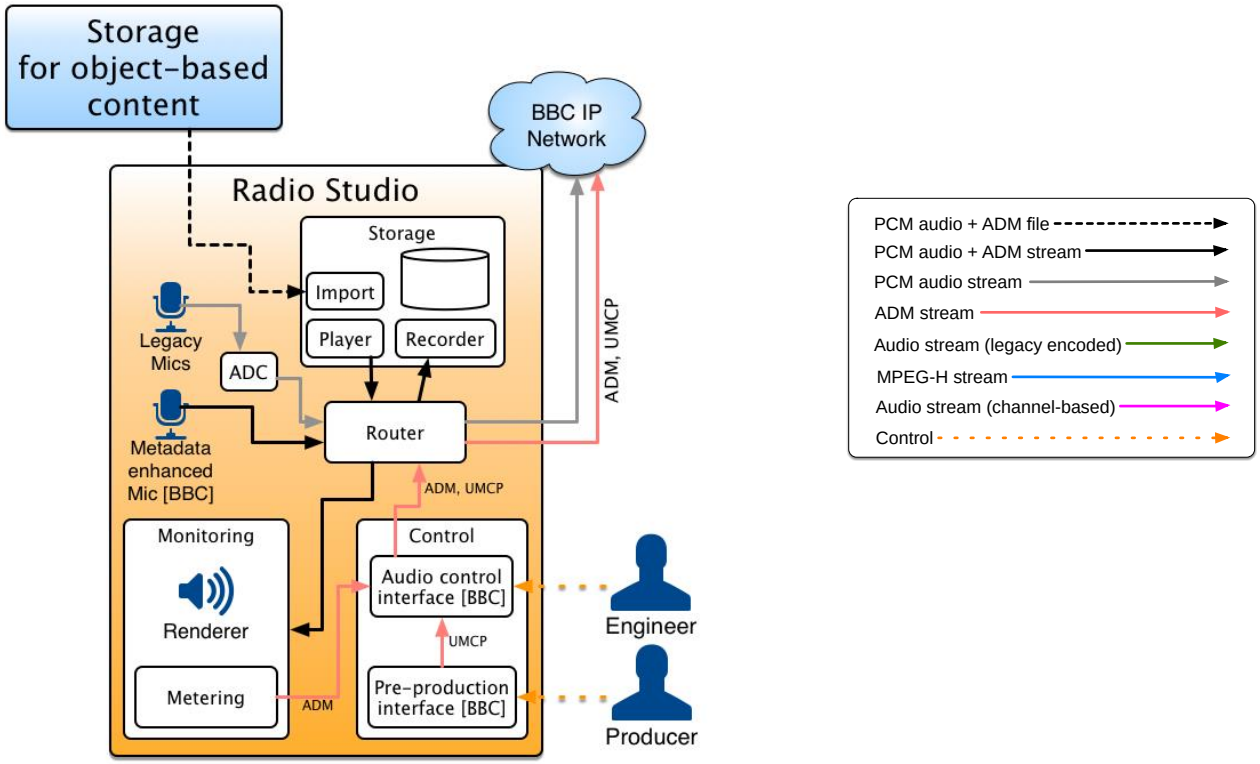

Figure 5: Current status of Radio Studio macroblock

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 output audio source. However, with the object-based approach the audio sources remain separate with generated additional metadata describing their relationship and how the renderer in the receiver has to combine them.

So, for describing a composite, the following information must be captured, stored and transmitted alongside the pure audio signal:

- **Sources**: Which audio sources are part of the composite, and which audio source the metadata relates to.
- **Gain and position**: How much gain has to 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. microphone)

For our IP-based Radio Studio we have chosen to use the NMOS specifications. This provides open standards for mechanisms that handle identification, timing, discovery and registration. We are also using the Audio Definition Model (ADM) to describe audio metadata such as gain, position and labelling.

Audio control interface

To capture and transmit this metadata, there must be a user interface that the engineer or producer can use to monitor and control this. The features of this user interface must include the following functionalities:

- **Creating a new composite**: Generating a blank programme, which 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 identities of contributors is usually known in advance, so this information – referred to as "programme metadata" – should be captured in pre-production. This pre-production information could be captured using a specialised interface designed for the producer, who is in charge of editorial tasks of the programme. Examples of data to be captured using this interface include:

- **Programme title/description**: An overall view of the composite
- **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 and made ready available, then it must be triggered at the appropriate moment. This could be done manually through the audio control interface with the engineering pressing a button. Alternatively, this 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.

Where a script of a programme is available, this too may be imported during pre-production. In other circumstances, a live transcript might be created by sending the audio to a speech-to-text converter. One mechanism under evaluation is shown in Figure 2, where audio is sent to a Google service and text is returned.

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 still in its early stage. 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 composites, which describe 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 be carried as JSON according to the UMCP schema 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.

Audio description metadata, such as gain and pan, are represented in the ADM and, when needed, converted to the corresponding MPEG-H metadata fields. Additional metadata, such as program information, GPS locations, speaker identification and other semantic metadata will be transported over JSON using a time-synchronous event model. These JSON streams will be included in the MPEG-DASH MPD manifests as separate 'Adaptation Sets'. The JSON events can also include links to

external sources.

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

Audio monitoring

A renderer is a system that takes an object-based audio composite 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 to standardise a 'baseline' renderer, however this development falls outside of the timeline of the ORPHEUS pilot phase 1. For the pilot we will use a VBAP-based renderer, starting with the BBC's own implementation, followed by the MPEG-H rendering system.

Audio metering

Metering involves processing an audio source using an algorithm to generate a meter level. The algorithms that we will use in the ORPHEUS pilot are loudness and true peak, as described by EBU R128. Each audio object will be routed through a processing node to generate the meter level values, and this will then feed the audio control interface. This will allow the engineer to monitor the audio signals of each object. The output of the studio renderer will also be metered to ensure the overall output levels are appropriate.

Recording/playback

The ability to record and replay material is an important 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', 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. This can later be replayed as if it were a recorded live stream. Later, we plan to implement a similar process so that we can export BW64+ADM files of content that was recorded live.

Interfaces

This macroblock imports object-based media as BW64+ADM files, and 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)
  These are a set of open specifications which define three important aspects of interfacing

object-based audio over IP:

- **Identity** – How to identify each individual audio source and stream of data ('flow') from those sources.
- **Timing** – How to define when audio was sampled, when metadata changed or when events occurred.
- **Discovery and registration** – How devices announce themselves as being available and advertise what streams they can offer.
- **Routing** – How streams get routed between devices on the network, and how devices can request to receive streams.

The NMOS specifications are available at http://nmos.tv

- **ADM** (Audio Definition Model)
  This is used as a data model as a standard way to describe a sound scene made up of audio objects. ADM is used in two ways in this pilot –firstly in combination with BW64 to describe object-based audio files, and secondly as a data model to describe audio parameters over UMCP for streaming of object-based audio. The ADM specification is described by ITU-R BS.2076.
- **UMCP** (Universal Media Composition Protocol)
  We are using UMCP to link the ADM metadata to NMOS-described audio sources. Currently, ADM is a file format and can only describe audio source as channels within the file it originated. In a live production, the audio objects are distributed on a network so cannot be described as 'channel 1', for example. 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 which 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. More information can be found in D4.2.
- **gRPC** ("Google" remote procedure call)
  One option being evaluated is the use of a Google cloud service for speech-to-text conversion. This would use the open source remote procedure call system "gRPC" (initially developed by Google) for sending audio and receiving text.

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

2.2.4 Distribution

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 as used within the broadcast infrastructure (AES67, ADM) into a more efficient format that is suitable for the transport over transmission channels and the reproduction in receiver devices (AAC, MPEG-H, DASH). The main distribution channel is the Internet but for backward compatibility also legacy systems such as DAB+ are considered. The specification and development of the distribution-macroblock is closely linked to the Tasks of Work Package 4 (Representation, Archive & Content Provision), especially T4.1 (Codecs and Formats) and T4.3 (Delivery to the end user).

The distribution-macroblock receives its input from the studio-macroblock via a private IP network as illustrated in Figure 6. The object-based audio is received as PCM via AES67 plus an ADM-based metadata stream. Both macroblocks use UMPC as the underlying protocol to establish and control data streams between each other (including audio and metadata). As the input interfaces and UMPC are already described in the previous subsection, we focus on the modules and output-interfaces of

the distribution-macroblock in the following.

For Pilot-1 two distribution paths are supported: An AAC-based distribution to HTML5 browsers and one for clients with support for MPEG-H 3D Audio. 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. 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.

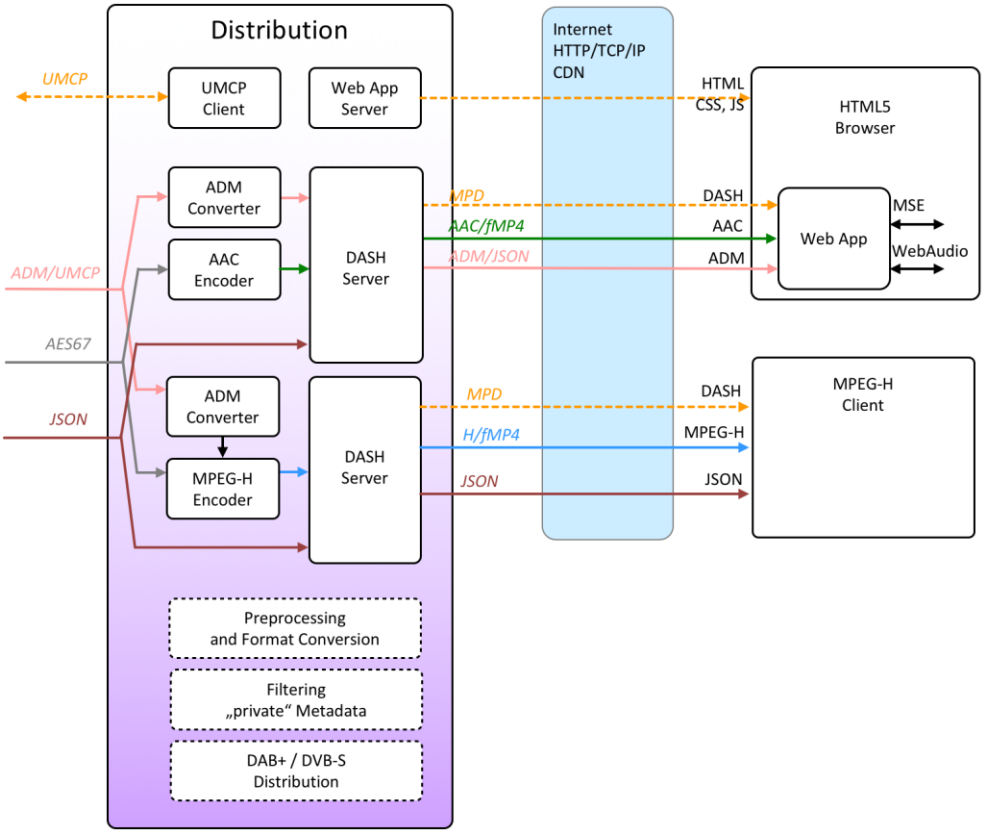

Figure 6: Scope of Distribution-Macroblock for Pilot 1

Distribution to HTML5 browsers

In order to reach a broad audience it is desirable to support commonly available browsers as reception devices for Pilot-1. 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 JavaScript is automatically downloaded and executed by the browser. It then downloads the Media Presentation Description (MPD) to start a DASH stream. The MPD links to the actual media, which is then downloaded as a series of fragmented MPEG-4 (fMP4) file segments.

There are then two approaches to decoding the fMP4 segments.

For standard channel layouts (stereo, 5.1) audio can be decoded using the Media Source Extensions (MSE) in the browser.

For higher channel counts and multiple streams the Web Audio 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. Decoded audio is passed through a JavaScript rendering engine (again using the Web Audio API) e.g. to generate a binaural stereo signal from multiple audio objects.

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 JSON-encoded ADM-metadata. 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, HTML, MSE, Web Audio API) and are therefore mentioned here.

It is worth noting that the exact definition of the streaming protocol does not have to be defined 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 MSE API and that the provided web application uses the MSE and WebAudio API correctly.

With respect to Figure 6, the following functionality is implemented in the macroblock for the distribution path to browsers: First, the ADM metadata has to be received via UMCP and interpreted, e.g. to configure the AAC-Encoder with the appropriate number of objects. The AES67 stream is received and the de-capsulated 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 who is also responsible for generating the MPD. In addition, the file segments with JSON-encoded ADM-metadata are generated, which are also streamed via DASH as a parallel data stream.

Distribution to clients supporting MPEG-H 3D Audio

Though the browser-based distribution is a flexible solution, which is immediately deployable, it has several shortcomings compared to a "true" NGA audio 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 of which 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 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. For Pilot-1 ORPHEUS follows the *Profile and Level,* which 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 shall also apply to Pilot-1. 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, Pilot-1 shall follow 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].

With respect to Figure 6, the following functionality is implemented in the macroblock for the distribution path to clients that support MPEG-H 3D Audio: First, the ADM metadata has to be received via UMCP 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 is received and the de-capsulated PCM audio is fed into the MPEG-H-Encoder. In addition, any 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. Additional metadata created in the radio studio or during the pre-production, such as program information, GPS locations, speaker identification and other semantic metadata will be transported over JSON using a time-synchronous event model. These JSON streams will be included in the MPEG-DASH MPD manifests as separate 'Adaptation Sets'. The JSON events can also include links to external sources.

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 are planned for Pilot-1: An iOS-App to be executed on an iPhone and an AVR.

For further study:

The given time frame for Pilot-1 requires setting priorities and focusing on the most essential functionality for distributing object-based audio. Additional functionality is scheduled with lower priority but still investigated within the scope of ORPHEUS and Pilot Phase 1. The functionality which is not considered essential for Pilot-1 is marked with dashed lines in Figure 6 and listed in the following.

Distribution to legacy devices via DAB+ and/or DVB-S: This additional distribution path requires the rendering of object-based audio to channel-based versions, e.g. as a simultaneous downmix into 2.0, 5.1 surround and binaural. In addition alternative language versions could be mapped into audio sub-channels and a subset of the audio-metadata could be mapped into existing metadata models of legacy systems.

Re-distribution and exchange: As for traditional content, the exchange and re-distribution of object-based content between different broadcasters is of practical interest. To verify feasibility it is intended to stream object-based audio over existing IP networks from the BBC London to BR Munich, implement above mentioned processes and air the pilot program live using a DAB+ test-transmitter at IRT Munich.

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.

Pre-Processing for format conversion: Depending on the capabilities of the distribution format (e.g. MPEG-H), the produced/archived audio content and metadata may needs 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. 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. Though a basic version for metadata conversion is needed for Pilot-1 (see also Figure 6), more advanced methods are for further study. Such a basic version is based on the "smallest common denominator" between ADM and MPEG-H, i.e. a restriction to features for which a direct mapping can be specified.

Pre-Processing for bit-rate and / or end-device adaptation: In order to meet the available bandwidth, the pre-processing component should also provide different versions of the content. This could be done either with reduced encoding qualities (e.g. spending only 48 kbit/s per object instead of 64 kbit/s) transmitted via MPEG-DASH, or with a pre-rendering of the audio scene where the total number of audio objects is reduced. This pre-rendering of the audio scene will take into account semantic information of the objects such as "importance" as well as the actual end-device capabilities.

Interfaces to other components

The distribution-macroblock has the following output-interfaces to the reception-macroblock:

Distribution 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

[MSE] Media Source Extension, W3C Recommendation 17 November 2016, https://www.w3.org/TR/media-source/

[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

[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

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

2.2.5 Reception

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.).

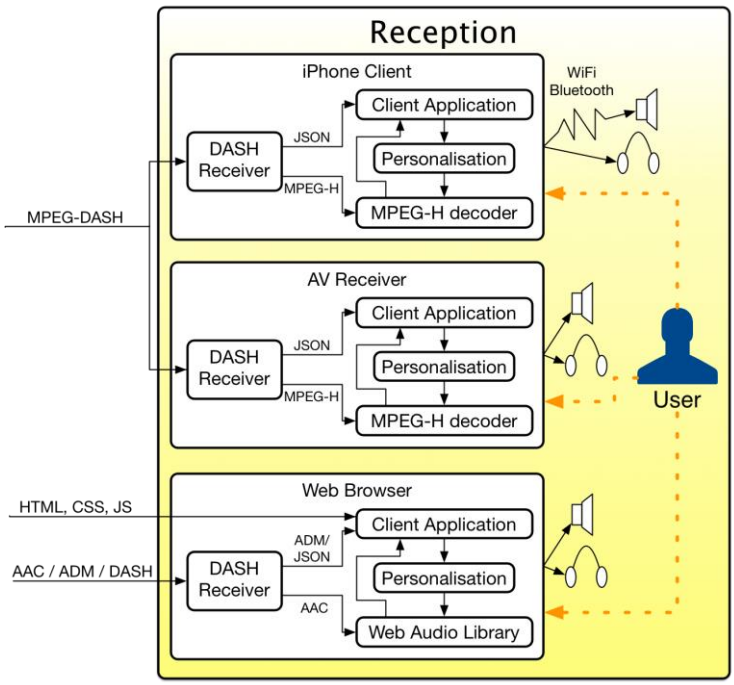

Figure 7: The three different implementations of the Reception macroblock

All solutions are comprised of two major components, a **decoding/rendering module** and a **personalisation module (**Figure 7**)**. 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 as well as ad-hoc multi-channel loudspeaker setups).

The personalisation module provides means for displaying and altering received metadata related either to mixing/spatial features as well as semantic properties. For instance, the user interface provides 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, name of panellists as well as live text transcription. When applicable, it provides also means for selecting the preferred language.

Metadata is received by two means: embedded in the MPEG-H or ADM stream (mostly for metadata related to the rendering of audio objects, such as location in space, volume, etc.) and in a separate JSON stream, that is added to the MPEG-DASH MPD file. The JSON stream contains timed events, such as textual transcripts, program information and possibly link to external sources (e.g. images). The personalisation can be applied to or based on information from both of these streams.

Optionally, an 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 device

Smartphones and tablets have become the most dominant devices for media consumption. Here, it is common to install custom apps instead of using the browser as it allows for the direct integration with the operating system. Therefore, also more efficient renderers can be implemented. An iOS app to be executed on an iPhone is being designed and developed by 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.

The reception macroblock for the mobile phone device is depicted on Figure 8. It receives MPEG-H 3D audio streams and optional additional JSON metadata over MPEG-DASH (See 2.2.4). The decoder/renderer of the MPEG-H client implements the Low Complexity Profile Level 3 of MPEG-H standard. In its current implementation, it is limited to a maximum of 16 simultaneously active (=rendered) signals out of a total of up to 32 channel, objects, and/or HOA signals in the bitstream as specified in the standard.

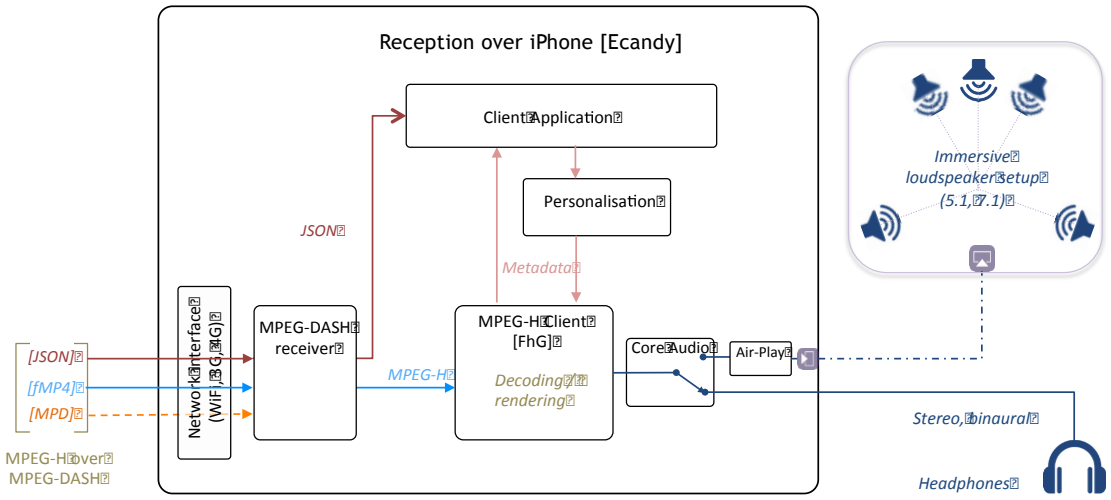

Figure 8: Mobile phone Reception Macro-block for Pilot 1

The main audio output format supported is binaural over headphones, or conventional stereo. Immersive audio playback over loudspeakers such as 5.1 or 7.1 is also made available via the Airplay protocol that is supported by a large number of hi-fi receivers.

The personalisation module implements the main functionalities described in the deliverable 5.1 (End-user requirements). The following personalisation features will be implemented in Pilot 1:

Programme Navigation:

- *Programme pause/resume;*
- *Jump to points of interest in the programme;*
- *Language option selection.*

Audio Quality and Rendering:

- *Output device/format selection;*
- *Foreground/background mix adjustment;*

Audio hardware device

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 object-based audio features provided in ORPHEUS will be added into the signal processing flow.

The reception macroblock for the audio-video device is depicted on Figure 9. It receives MPEG-H 3D audio streams and optional additional JSON metadata over MPEG DASH (See 2.2.4). As for the mobile device, the decoder/renderer of the MPEG-H client implements the Low Complexity Profile Level 3 of

MPEG-H standard. In its current implementation, it is limited to a maximum of 16 simultaneously active (=rendered) signals out of a total of up to 32 channel, objects, and/or HOA signals in the bitstream as specified in the standard. For the pilot phase 1, only audio related metadata will be handled; these metadata are embedded in the MPEG-H stream and transmitted to the client app through the MPEG-H client, thus the optional additional JSON metadata will not be used.

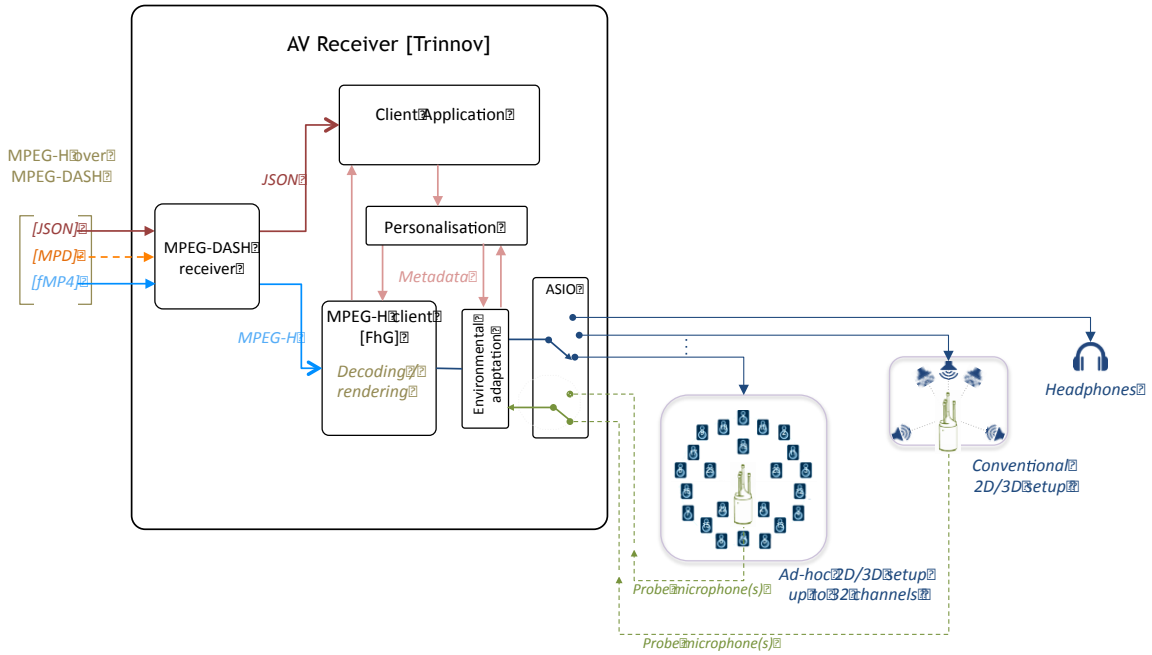

Figure 9: Reception macroblock for the AV Receiver (Pilot 1)

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

The personalisation module implements the main functionalities described in the deliverable 5.1 (End-user requirements). The following personalisation features will be implemented in Pilot 1:

Programme Navigation:

- *Programme pause/resume;*
- *Jump to points of interest in the programme;*
- *Language option selection.*

Audio Quality and Rendering:

- *Foreground/background mix adjustment:* An interactive GUI allows the user to adjust the balance between the foreground and the background sound. Actually, only the background level is adjustable in the MPEG-H API, but the loudness adjustment ensuring that the global level remains constant makes this mix behave naturally, with one slider to select the prominence of the foreground over the background.
- *Dynamic range compression profile*: Dynamic Range Control (DRC) metadata is a set of time-varying gain values that is generated and encoded by the MPEG-H audio encoder. DRC metadata allows for receiver side adaptation of the original audio content to different target devices and listening conditions amongst others. Currently, 5 presets of DRC are proposed (“None”, “Late night”, “Noisy environment”, “Dialog enhancement”, “General compression”)

Environmental adaptation

The receiver is a PC-based architecture, allowing sufficient CPU power to perform advanced processing for the environmental adaptation. The correction is applied at the last stage of the processing flow, to make the setup as close as possible to the ideal one awaited by the decoder and renderer. Thus, complex interactions between the renderer and the environmental adaptation process are not needed.

- *Time and level alignment*

  The loudspeakers positions are measured with the acoustic probe. They are time and level aligned to ensure a correct rendering, making them virtually equidistant from the sweet spot.

- *Equalisation*

  An automatic equalisation is performed, with different filter types (IIR, FIR). The target equalisation is specified in the GUI, and can differ from a flat one.

- *Room and speaker compensation*

  The loudspeakers and room acoustic are measured and corrected, to ensure a room and setup independent rendering. Multiple measures are merged to allow a larger good listening area.

Web browser

Today, a 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 plugins, installing apps, configuring speaker setups etc.). At present, web browsers do not readily support object-based audio rendering. BBC and IRT are working on the implementation of a rendering system based on JavaScript and the Web Audio API. 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 for the Web Browser is depicted on Figure 10. It receives AAC audio streams JSON-encoded ADM metadata over DASH (See 2.2.4). The DASH receiver decodes the encapsulated AAC bit-streams using the Web Audio API decodeAudioData() method. The multiple audio objects are rendered according to transmitted ADM metadata.

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

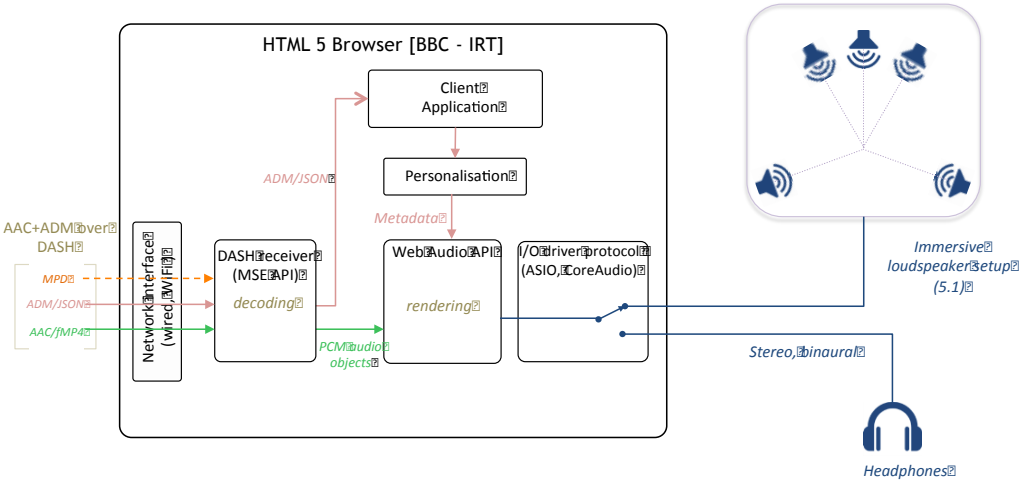

Figure 10: Reception macroblock for the Web Browser (Pilot 1)

The personalisation module implements the main functionalities described in the deliverable 5.1 (End-user requirements). The following personalisation features will be implemented in Pilot 1:

Programme Navigation

- *Programme pause/resume button*
- *Jump to points of interest in the programme*
- *Language option selection*

Audio Quality and Rendering

- *Foreground/background mix adjustment*
- *Dynamic range compression profile*

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

3 Integration

This section details the current status of the integration activities for the first pilot phase.

The consortium agreed to follow internal milestones for the integration where tasks or functionalities are defined. These internal milestones started with very little complexity to test the very basic features such as import and export of one BW64 (Broadcast Wave 64) file to and from the DAW (Digital Audio Workstation). Consequently, the complexity will increase with later milestones. The subsections below provide an overview of the activities and achievements.

3.1 Integration Activities

Internal Milestone	Action	Status
#1	A: Multiple objects with gain and pan, rendered in studio [BBC, FHG]	Ongoing
#1	B: ADM gain and pan information viewable and editable in Sequoia automation [MAGIX]	Completed
#1	C: Import BW64+ADM file into IP Studio [BBC]	Completed
#1	D: Stream objects over AAC+DASH (without ADM) to Web Audio renderer (and iPhone?) [BBC, IRT, ECANDY]	Completed
#2	A: Multiple objects with gain and pan, encoded in MPEG-H, streamed over MPEG-DASH to Chromium and iPhone (and Trinnov AV receiver?) [BBC, FHG, ECANDY, TRINNOV]	Ongoing
#2	B: Mic array rendering, written to BW64+ADM file [BCOM, FHG, MAGIX]	Ongoing
#2	C: Stream objects over AAC+DASH+ADM to Web Audio renderer [BBC, IRT]	Ongoing

3.2 Planned milestones

Internal Milestone	Action	Status
#3	A: Multiple objects with gain and pan, rendered in studio [BBC, FHG]	Ongoing
#3	B: Stream objects over AAC+DASH+ADM to Web Audio renderer [BBC, IRT]	Ongoing
#3	C: Multiple objects with gain and pan, encoded in MPEG-H, streamed over MPEG-DASH to Chromium and iPhone and Trinnov AV receiver [BBC, FHG, ECANDY, TRINNOV]	Ongoing
#3	D: Recruit production team to create pilot	Ongoing
#3	E: Prepare for interaction features in milestone 4	Ongoing
#4	A: Mini-pilot	Ongoing
#4	Pilot production planning	Ongoing

4 Conclusions

To specify a reference architecture for an end-to-end infrastructure for object-based audio, the ORPHEUS consortium defined a pilot implementation architecture to be used for the dedicated pilots during the project. This pilot implementation architecture will serve as basis for the reference architecture by taking into account findings and lessons learned during the pilot phases. The defined pilot architecture provides specific solutions, formats and interfaces which suit the requirements of ORPHEUS and its partners best. Specific subsets or profiles of identified are mentioned and described more detailed in D4.2.

To realize the aimed infrastructure, the integration of technical components developed in WP3 – 5 with existing infrastructure and tools has started and is ongoing to the integration plan presented in sections 3.2 and 3.3.

For a successful pilot phase, the integration of the developed tools and components is crucial for the next months of ORPHEUS. Following a stepwise approach with internal milestones that require more and more complex tasks, the ORPHEUS partners are confident to match the envisaged goal in timely manner.

This document will be updated again in project month M27 (D2.4), after gathering experience on using the implemented architecture within the pilot 1, targeting extension of the architecture for the pilot 2.

References

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

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

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

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

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

[6] ITU-R Recommendation BS.2076, “Audio Definition Model”

[7] ITU-R Recommendation BS.2094, “Common Definitions for the Audio Definition Model”

[end of document]