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

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

D4.2: Interim report on the work on representation archiving and provision of object-based audio

Version: v0.7

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	31/03/2017
Submission date	12/05/2017
Lead editor	Andreas Silzle (FHG)
Authors	Nikolaus Färber (FHG), Michael Meier (IRT), Tilman Herberger (Magix), Andrew Mason (BBC), Chris Baume (BBC), Matt Firth (BBC), Matt Paradis (BBC)
Reviewers	Werner Bleisteiner (BR)
Work package, Task	WP 4, T4.1, T4.2, and T4.3
Keywords	Formats, BW64, ADM, NMOS, UMCP, MPEG-H, DASH, IP-Studio, Sequoia, iOS app, browsers

Abstract

This deliverable describes the progress on representation, archiving and provision of object-based audio. It builds on D4.1 “Requirements for representation, archiving and provision of object-based audio”. It lists the formats which are selected for ORPHEUS and describes the interim status of the implementation of these formats. On the production side, formats like BW64, ADM, NMOS, UMCP are used and explained. BW64 is also used for archiving. For provision or distribution MPEG-H and AAC + ADM metadata are selected. Both solutions use MPEG-DASH for streaming.

This Deliverables also serves as documentation for milestone MS12 “Initial implementation and documentation of a format for provision of objected-based audio” which has been achieved on 31/03/17.

Document revision history

Version	Date	Description of change	List of contributor(s)
0v1	28/03/2017	Initial version	Andreas Silzle (FHG), Nikolaus Färber (FHG)
0v2	10/042017	Merge of input from IRT, Magix, BBC, FHG	Michael Meier (IRT), Tilman Herberger (Magix), Andrew Mason (BBC), Chris Baume (BBC)
0v3	11/04/2017	Added UMCP description	Chris Baume (BBC), Matt Firth (BBC), Andrew Mason (BBC)
0v4	12/04/2017	Added AAC+ADM distribution description, and IP Studio implementation overview	Matt Paradis (BBC), Chris Baume (BBC)
0v5	19/04/2017	MPEG-DASH, MPEG-H streaming	Nikolaus Färber (FHG)
0v6	26/04/2017	Archiving, metadata translation	Michael Meier (IRT), Nikolaus Färber (FHG)
0v7	12/05/2017	Final editing and submission	EURES

Disclaimer

This report contains material which is the copyright of certain ORPHEUS Consortium Parties and may not be reproduced or copied without permission.

All ORPHEUS Consortium Parties have agreed to publication of this report, the content of which is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License[1].

Neither the ORPHEUS Consortium Parties nor the European Commission warrant that the information contained in the Deliverable is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using the information.

Copyright notice

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

Executive Summary

This deliverable describes the progress on representation, archiving and provision of object-based audio. It builds on D4.1 “Requirements for representation, archiving and provision of object-based audio”.

This deliverable lists the formats which are selected for the ORPHEUS project and describes their interim status of implementation. On the production side formats like BW64, ADM, NMOS, UMCP are selected, used and explained. BW64 is the main format for audio and metadata. It allows file sizes larger than 4 GByte and includes the metadata in ADM format. The within the ORPHEUS project necessary and selected metadata parameters are nominated and explained, including their respective value range. NMOS and UMCP are described as fundamental formats in BBC’s IP Studio production software. For UMCP an example is given in Appendix A.

Various ongoing standardisation activities for these formats, in which several partners are involved, are mentioned. The implementation status of the formats in IP Studio and the DAW Sequoia are explained as well as practical issues for ADM to MPEG-H metadata conversions and the requirements for archiving using BW64.

For provision or distribution to the end-user MPEG-DASH is used for streaming to HTML5 browsers or the MPEG-H clients. The used standards and the technical details for MPEG-DASH and the AAC plus ADM Streaming are listed, as well as the streaming solution of MPEG-H over DASH. For AAC plus ADM streaming the necessary steps are encoding, segmentation, transfer to public domain, media reference and playback and rendering. The advanced features of the Next Generation Audio (NGA) codec MPEG-H are mentioned, including the way how packaged into MPEG-DASH. Short syntax examples are given for both cases. The delivered software packages and the status of the implementation at the different partners are described, for the browser, AV receiver and the iOS mobile application.

Several references to the used standards and explaining publications are given.

This Deliverables also serves as documentation for milestone MS12 “Initial implementation and documentation of a format for provision of objected-based audio” which has been achieved on 31/03/17.

Table of Contents

List of Figures

List of Tables

Abbreviations

ADM	Audio definition model
AES67	Standard for audio-over-IP interoperability
ASI	Audio Scene Information
BBC	British Broadcasting Corporation
BR	Bayerischer Rundfunk
BW64	Broadcast Wave 64Bit
BWF	Broadcast wave
Dante	Digital Audio Network Through Ethernet
DASH	Dynamic Adaptive Streaming over HTTP
FHG	Fraunhofer Gesellschaft
HOA	Higher Order Ambisonics
IP	Internet Protocol
IPF	Instantaneous Playout Frames
ITU-R	International Telecommunication Union, Radiocommunication Sector
JSON	Format for exchanging object data
Livewire	Audio-over-Ethernet system
MHAS	MPEG-H Audio Stream
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
NGA	Next Generation Audio
NMOS	Networked Media Open Specifications
OB	Outside broadcasting
PCM	Pulse-code modulation
Q-LAN	Audio over IP audio networking technology component
RAVENNA	Technology for real-time transport of audio and other media data
RIFF	Resource Interchange File Format
SMIL	Synchronized Multimedia Integration Language
UMCP	Universal Media Composition Protocol
XML	Extensible Markup Language

1 Introduction

This documents builds on deliverable 4.1 “Requirements for Representation, Archiving and Provision of Object-based Audio”. It lists the formats which are selected for ORPHEUS and describes the interim status of the implementation of these formats for representation, archiving and provision of object-based audio. On the production side most of the formats are used, like BW64, ADM, NMOS, UMCP, see section 2. For the provision or distribution MPEG-H and AAC + ADM metadata streamed over MPEG-DASH are used, see section 3. The implementation and usage of the different formats in ORPHEUS and their capabilities are explained.

2 Production Formats for Distribution and Archiving

2.1 Broadcast Wave (BW64)

The Broadcast Wave (BW64) format is the successor of Broadcast Wave Format (BWF) [1], the standard audio data file format used today in broadcast productions. The BW64 format allows for maximum file sizes larger than 4GB by using 64Bit signalling and is therefore also capable of transporting immersive audio and a large number of objects. BW64 contains additionally a chunk for the ADM metadata.

Recommendation ITU-R BS.2088 [2] “Long-form file format for the international exchange of audio programme materials with metadata” is referred to by the abbreviation “BW64” for “Broadcast Wave 64Bit”. The BW64 file is a type of Resource Interchange File Format (RIFF) and includes “chunks” <ds64>, <axml> and <chna> that enable the file to carry large multichannel files and metadata.

The metadata includes the Audio Definition Model (ADM) specified in Recommendation ITU-R BS.2076 [3].

The <ds64> chunk provides the larger file size indicator, to allow files bigger than 4 GBytes.

The <axml> chunk is defined for storing and transferring metadata as XML. This is where the ADM XML metadata will be contained for file-based uses. It can also store non-ADM metadata should it be required.

The <chna> chunk provides the references from each track in a BW64 file to the IDs in the ADM metadata defined in Recommendation ITU-R BS.2076.

While the BW64 can handle a variety of different audio sample formats, in ORPHEUS the format will be:

- 24-bit signed PCM
- 48kHz sample-rate

BW64 with ADM metadata stored in a <axml> chunk has now been implemented in the Magix Sequoia DAW, and in IRCAM’s ADM tools. This format is being used for the exchange of programme material between the project partners, as well as for local storage.

2.2 Audio Definition Model (ADM)

The most recently defined and most complete metadata model that supports the description of object-based audio (as well as channel-based and scene-based audio) is the “Audio Definition Model” (ADM), which was selected for use in the ORPHEUS project.

The Audio Definition Model (ADM) is specified in ITU-R BS.2076 [3]. The ADM defines the structure of a metadata model that allows the format and content of audio files to be reliably described. It specifies how XML metadata can be generated to provide the definitions of tracks in an audio file. Its flexibility allows is to indicate channel, object and HOA-based audio, with no limits on size or complexity of the audio.

Another ADM-related ITU document is ITU-R BS.2094 [4], which describes a list of commonly used definitions for channel-based audio (common definitions for HOA-based audio are soon to be released). These common definitions remove the need to denote different configurations explicitly in files, as they can be simply cross-referenced.

File-based ADM

The specification of the ADM in BS.2076 uses XML as the description language, and is targeted at file-based applications, in particular the BW64 file format (see Section 2.1). This means the audio and its associated metadata is treated as a single complete continuous file. This works fine for exchange of complete programmes, but is not suitable for streaming (particularly in live scenarios) where audio and metadata is being generated and distributed in real-time.

Serial ADM

Currently, the ITU (Rapporteur Group WP6B-RG13 in ITU-R) are developing a serialised (frame-based) version of the ADM, which will allow for streaming and real-time applications. Although more than one approach is currently under consideration, it will be compliant with the fundamental metadata model described in BS.2076. So all the parameters in the file-based version will be available in the serialised form.

Parameters for ORPHEUS

As the ADM contains a wide range a parameters and possible configurations, it would be very easy to generate ADM metadata that becomes difficult to process and transcode. Therefore, to ensure the ADM metadata is manageable in the ORPHEUS project, a subset of parameters needs to be defined. Each of the ADM elements is covered in the following subsections. It is assumed that all the ID, naming, ID-referencing, and timing-related attributes and sub-elements will be used.

audioBlockFormat

The majority of the parameters reside within the *audioBlockFormat* element (which is a child of the *audioChannelFormat* element), and can cover all the different types of audio (channel, object, HOA, etc.). Some of these parameters are for very specific scenarios, which are unlikely to be required in this project, and will thus be ignored.

The ADM allows different types of audio to be specified using the *typeDefinition* attribute. For ORPHEUS we limit ourselves to the two types in Table 1.

Type of audio	typeDefinition
Object-based	“Objects”
Channel-based	“DirectSpeakers”

Table 1: ADM typeDefinitions

For “Objects” type of *audioBlockFormats*, only the sub-elements in Table 2 should be used.

Sub-element	Attributes	Possible values	Default
position	coordinate=”azimuth”	-180.0 to 180.0	
	coordinate=”elevation”	-90.0 to 90.0	
	coordinate=”distance”	0.0 to inf	1.0
gain		0.0 to inf	1.0
diffuse		0.0 to 1.0	0.0
jumpPosition		0 or 1	0

Table 2: audioBlockFormat elements for "Objects"

For “DirectSpeakers” type of audioBlockFormats, only common definitions as specified in [4] are to be applied.

audioChannelFormat

As *audioChannelFormat* is the parent of audioBlockFormat, it contains no other sub-elements apart from ‘frequency’ which is covered by the common definitions.

audioPackFormat

As *audioPackFormat* is for grouping channels it references a list of *audioChannelFormats* and other *audioPackFormats* (for nesting purposes), so it does not contain many other parameters. The only parameter it carries is shown in Table 3, and can be used in ORPHEUS.

Sub-element	Attributes	Possible values	Default
absoluteDistance		0 .0 to inf	1.0

Table 3: audioPackFormat element

audioObject

This element connects the audio essence with its format. It also contains sub-elements and attributes related to interactivity. The attributes and sub-elements used in ORPHEUS are shown in Table 4.

Sub-element	Attributes	Possible values	Default
Top Level	dialogue	0,1 or 2	2
Top Level	interact	0 or 1	0
audioComplementaryObjectIDref		AO_xxxx (ID string)	*N/A*
audioObjectInteration		Contains sub-elements, all apply.	*N/A*

Table 4: audioObject elements

audioContent

The element describes the content of a particular object. The attribute and sub-elements used for ORPHEUS are shown in *Table 5*.

Sub-element	Attributes	Possible values	Default
Top Level	audioContentLanguage	Two-letter language string	*N/A*
dialogue		0,1,2	2
	nonDialogueContentKind	0,1,2	0
	dialogueContentKind	0,…,6	0
	mixedContentKind	0,…,3	0

Table 5: audioContent elements

audioProgramme

The element describes the whole programme. The attributes used for ORPHEUS are shown in *Table 6*.

Sub-element	Attributes	Possible values	Default
Top Level	audioProgrammeLanguage	Two-letter language string	*N/A*

Table 6: audioProgramme elements

audioTrackFormat and audioStreamFormat

As the assumed audio format will be PCM, this makes use of the *audioTrackFormat* and *audioStreamFormats* straightforward. There would be a one-to-one connection between the two elements, with the *formatDefinition* attribute set to "PCM". The *audioStreamFormat* element would also use a single *audioChannelFormatIDRef* reference sub-element.

2.3 EBUCore

The "EBUCore" set of metadata defined in EBU Tech 3293 [5] has been identified as being the minimum information needed to describe radio and television content. It is a collection of basic descriptive and technical/structural metadata elements for audio-visual content and an extension of the Dublin Core metadata model[2]. It is directly compatible with the EBU Class Conceptual Data Model Tech 3351 [6] leading to its compliant use in Semantic Web and Service Oriented Architectures.

The specification addresses the creation, management, and preservation of audio-visual material. It facilitates programme exchanges between broadcasters or between production facilities in distributed and cloud environments. Beyond production, EBUCore can be used to describe content for distribution (broadcast, broadband Internet, mobile, or hybrid delivery).

The Audio Definition Model (ADM) has been implemented in EBUCore as a new "audioFormatExtended" element. The EBUCore schema is kept strictly aligned with approved ITU changes to Recommendation ITU-R BS.2076.

EBU Tech. 3293 may be downloaded free of charge from the EBU.

It has not yet been decided to what extent EBUCore will be applied within the project.

2.4 AES67

For live production, object-based audio has to be streamed in real-time within the production workflow. The most common format for IP-based infrastructure is AES67, which was developed by the Audio Engineering Society and published in September 2013. AES67 is the "AES standard for audio applications of networks - High-performance streaming audio-over-IP interoperability".

There are high-performance media networks that support professional quality audio with low latencies. The level of network performance required to meet these requirements is available on local-area networks and is achievable on enterprise-scale networks. Before AES67 was published, there were no recommendations for using in an interoperable manner, the multiplicity of networked audio systems that had been developed. It is a layer 3 protocol suite based on existing standards and is designed to allow interoperability between various IP-based audio networking systems such as RAVENNA, Livewire, Q-LAN and Dante. BBC R&D was actively involved in the development of the standard within the AES Standards Committee.

AES67 provides comprehensive interoperability recommendations in the areas of synchronization, media clock identification, network transport, encoding and streaming, session description, and connection management. It does not in itself define any new protocols. Instead, it provides specific recommendations for interoperability. It defines how existing protocols are used to create an interoperable system.

[2] http://dublincore.org/

The AES67 description may be purchased from the Audio Engineering Society (free of charge to members thereof).

BBC R&D's IP Studio platform uses a very similar scheme to AES67 to achieve low-latency transport and synchronisation of media. Both use Realtime Transport Protocol (RTP) with timestamps derived directly from a common network-distributed reference clock. As such it is straightforward to integrate commercial audio products that implement AES-67 to provide direct IP-based I/O.

The IP Studio implementation adds some additional timing and identity metadata to the streams, but these are "piggy-backed" using RTP header extensions as defined by the RTP suite of standards, so compatibility with RTP as used in AES-67 is preserved.

A PTP grand master clock is connected to the IP Studio network that has been set up for the ORPHEUS project. It is currently synchronized by NTP to BBC R&D's network time servers; it is planned to synchronise it to GPS, but the infrastructure (and structure) of the building still present some challenges.

The AES67-compliant devices on the IP Studio network will be managed using NMOS (see section 2.5), where this can be incorporated into the device. Devices developed by project partners (such as Magix Sequoia) might be augmented with this, but "off the shelf" products being used, (such as the Axia XNode), might not.

2.5 NMOS

The Networked Media Open Specifications (NMOS) are a family of specifications, produced by the Networked Media Incubator (NMI) project by the Advanced Media Workflow Association (AMWA), related to networked media for professional applications. At the time of writing, NMOS includes specifications for:

- Stream identification and timing
- Discovery and registration
- Connection management

A full explanation of NMOS and how it is being used in ORPHEUS can be found in Section 2.1 of Deliverable 3.4 ("Implementation and documentation of a live object-based production environment").

2.6 UMCP

The Universal Media Composition Protocol (UMCP) is a method to describe the assembly and processing of media sources/assets to produce a new piece of media. A UMCP Composition is the package of instructions required to do this.

Within a typical production workflow, there are likely to be multiple contributing media sources, and each source may go through various different processors. For example, an audio feed may be attenuated or amplified and subsequently panned, or a video feed may be cropped, resized and repositioned. UMCP is used to describe and control these pipelines. These pipelines generally form an inverted tree structure, culminating in the Composition output. In technical terms, they form a directed acyclic graph.

Every node (or stage) of the media processing graph is defined by a UMCP Sequence. A Sequence may control the contribution of media sources in to a pipeline, define and control a processing stage, or define the ultimate output point of the Composition. The UMCP Composition contains timed Events to control these Sequences. An event is essentially a collection of instructions for a precise moment in a Composition timeline. These instructions could control processor parameters or start/stop the feed from a media source. Event instructions are also used to define the interconnections between nodes to build the processing graph. Since Events can be defined at any point on the Composition timeline, it is possible to build and modify the processing graph dynamically over time.

2.6.1 Foundational Technologies

IP Studio is a BBC R&D project to investigate and capitalise on the advantages that an IP production infrastructure can offer. UMCP uses the software framework developed by the IP Studio project as a backbone. In particular, it uses the IP Studio Media Access API for retrieving sources of media and for advertising its own published Compositions.

UMCP is authored, stored and retrieved via an API. The API holds a database of Compositions and the Events belonging to each Composition. Since these Events are wrapped in a data format utilised by IP Studio, they can be handled within the IP Studio infrastructure.

2.6.2 Data Model

UMCP is built on the NMOS (Networked Media Open Standard) Content Model, which is also at the heart of the IP Studio framework. In this model, audio and video streams are divided into very small segments known as ‘grains’. Each grain carries identifiers and timestamps describing where it came from and when it was captured in relation to a common reference clock. These grain attributes are carried with the media payload as it’s streamed, and stored with the grains when recorded. Using the grain identity and timing attributes, IP Studio client applications can reference live sources in the same way as they refer to stored content and synchronise content from different streams according to their original timing relationships

As well as audio and video, grains can also be used for storing and transporting control data or metadata. This is the underlying mechanism upon which UMCP is based; UMCP Events are encapsulated within NMOS data grains. These data grains form streams of Events on the composition timeline, making UMCP appropriate for live production as well as post-production.

In fact, UMCP uses only data grains - it does not provide any audio or video essence itself. Media sources exist within IP Studio independent of a Composition. UMCP simply references them using their unique identities. A good analogy is that UMCP is a cookbook – it provides recipes, but not the ingredients or utensils. It also does not do any of the cooking for you – it simply provides comprehensive instructions on what you should do and when you should do it in order to create the intended end product. It is this ‘recipe’ notion that enables UMCP to support object-based productions.

2.6.3 Working Principle

Clients ‘subscribe’ to a Composition by opening a WebSocket connection to the API server with a specific ID. Using this communications channel, the client will receive any changes made to the Composition in real-time in the form of UMCP Events. The client may also request their own changes to the Composition using this channel. If their request is successful, the client will be notified by way of new Events sent back from the API, as will all other connected clients. Upon failure, only the client who made the request is notified. This is because no Composition changes are made and the failure message is therefore insignificant to the other clients.

This system allows multiple users to collaborate upon a single Composition concurrently. This consequently means that UMCP can be used for live production. For example; an edit terminal would create and subscribe to a Composition, and then request live edits in real-time. A rendering engine would be subscribed to the same Composition and would therefore receive these edit decisions as Events from the API. The resulting media can be rendered out accordingly for delivery across existing mediums. As well as receiving, validating and distributing these edit decisions as new Events, the UMCP API simultaneously records all of the Events in the Composition database.

UMCP also offers the ability to deliver a truly object-based experience to the viewer in future. By simply providing the Composition data direct to the end user, the Composition could be rendered out client-side according to the users' needs and/or the capabilities of their viewing device.

2.6.4 Example

To better illustrate the operation of UMCP, an example description can be found in Appendix A.

2.7 The Broadcast Metadata Exchange Format (BMF)

The Broadcast Metadata Exchange Format Version 2.0 (BMF 2.0)[3],[4] is a metadata model for the harmonization and standardization of metadata and the exchange thereof.

It is a long-term development by the IRT in close cooperation with German public broadcasters to provide a rich metadata vocabulary for domains of television, radio and online. The model includes and combines both editorial and technical metadata and will form the basis of many upcoming IT-based technologies used to exchange media and data between German public broadcasters.

While BMF is not used directly within the ORPHEUS pilot, interoperability and compatibility with this format is important to ease the adoption of object-based audio workflows by German public broadcasters.

IRT is currently exploring the extension of BMF by the ADM metadata set required to produce next-generation, object-based audio. Furthermore, strategies to ensure the compatibility of metadata that already exists in both models are explored.

This way it will be ensured that current technologies that are able to work with BMF can be easily extended to support object-based audio based on the ADM.

2.8 Standardization

The standardization of ADM is ongoing and the project members have a great influence for new and necessary features for object-based audio. More details about the standardization activities at ITU-R are described in a section 2.4 of deliverable 6.2 “Intermediate Standardisation and Dissemination Activity Report”.

The latest activity has been in ITU-R Working Parties 6B and 6C, in particular at the meeting cycle in March 2017. The Audio Definition Model (Recommendation ITU-R BS.2076) and its companion document of common definitions (Recommendation ITU-R BS.2094) have been enhanced and refined, to add a clearer specification of how to support high-order Ambisonics format signals. The draft revisions were approved by the Study Group at the end of the cycle, and will be sent to Administrations for approval and adoption.

A further document, Report ITU-R BS.2388 “Usage Guidelines for the Audio Definition Model and Multichannel Audio Files” was also updated at this meeting cycle, to reflect experienced gained from use of ADM in the real world. Notably, more guidance is now being given on the handling of

[3] http://bmf.irt.de

[4] https://www.irt.de/en/activities/data-and-security/metadata/

programme preambles (count-down clocks, etc.) in recordings, and on the choice of the appropriate size of block for dynamic objects.

Because the ADM is a crucial part of interoperability between systems, work continues in ITU-R to standardize one or more renderers of object-based audio, with the requirement that they use ADM metadata.

A standardized renderer is also seen as a pragmatic approach to loudness measurement (and thus loudness normalization) of object-based audio. The study in ITU-R on loudness measurement in this are awaits the work of the Rapporteur Group on the renderer(s).

2.9 Implementation

2.9.1 IP Studio

We are using BBC's IP Studio as a platform for our ORPHEUS pilot, as it provides a framework and set of tools based on open standards that we can use to implement our pilot architecture. IP Studio makes full use of the NMOS specifications for identity, timing, discovery and registration (see Section 2.5), and implements the APIs described in those specifications.

In this section, we discuss what has been implemented for audio, metadata and storage.

2.9.2 Audio

IP Studio has built-in support for streaming audio content over 'grains'. Each grain is an RTP packet with additional header information that contains identity and timing information. The payload of the packet is PCM audio. This can be in any format, but often uses 24-bit signed integers or 32-bit floating points.

RTP receiver and transmitter plugins have been implemented for connecting to AES67 sources and receivers. These can be dynamically reconfigured by using the NMOS Node API.

2.9.3 Metadata

Metadata can be streamed in IP Studio using 'event grains'. These are similar to audio grains in that each is an RTP packet with identity and timing information in the header, but instead of carrying a payload of PCM audio, they contain JSON strings.

The UMCP API has been implemented, and can be used to route and store control metadata linked to audio flows. A UMCP receiver plugin has also been created, which subscribes to a single UMCP composition and transmits a stream of event grains. This can be used to generate the stream of metadata for driving an audio renderer, or for broadcasting to the audience.

2.9.3.1 Storage

Audio and event grains can be routed to a 'sequence store' which saves and catalogues each grain in a database. These can then be queried and recovered using the Media Access API, which lists the data that is available, and can extract and package the grains that were recorded between two timestamps.

In order to support the BW64+ADM standard, we have developed an import script that converts the ADM metadata to a UMCP composition and ingests the data into a sequence store. We are currently working on an export script that can be used to generate a BW64+ADM file based on a recording in the sequence store.

2.9.4 Sequoia DAW

The Sequoia DAW has an object based approach from the beginning. This means, that real time audio effects processing can not only be assigned to tracks, busses and the master, but also to each audio clip individually, see Figure 1.

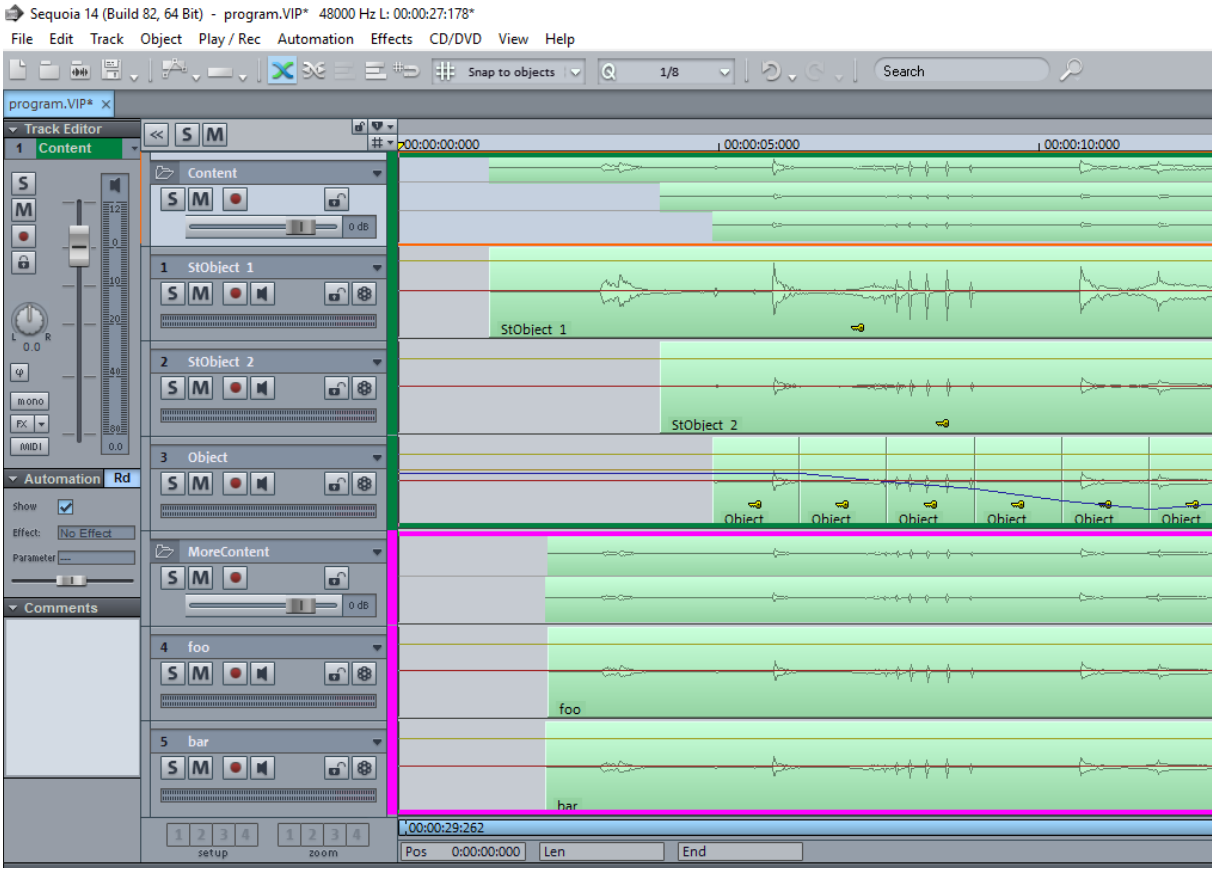


Figure 1: Single audio object tracks in Sequoia

Using this advantage MAGIX incorporated the following ADM features in Sequoia and made them editable in the GUI:

- **ADM Import / Export** using the file menu

menu option: File -> Save ADM File

- metadata only: creates separate xml file in project directory (for testing purposes)
- audio export: full export of audio + ADM

- **ADM program** – corresponds to a Sequoia VIP (virtual project = multi track arrangement, project title becomes ADM program name)

- **ADM content element** – corresponds to a Sequoia folder track, which may contain any number of tracks, see Figure 2.

- **ADM object** – corresponds to a Sequoia audio track, which may contain any number of clips, see Figure 3.

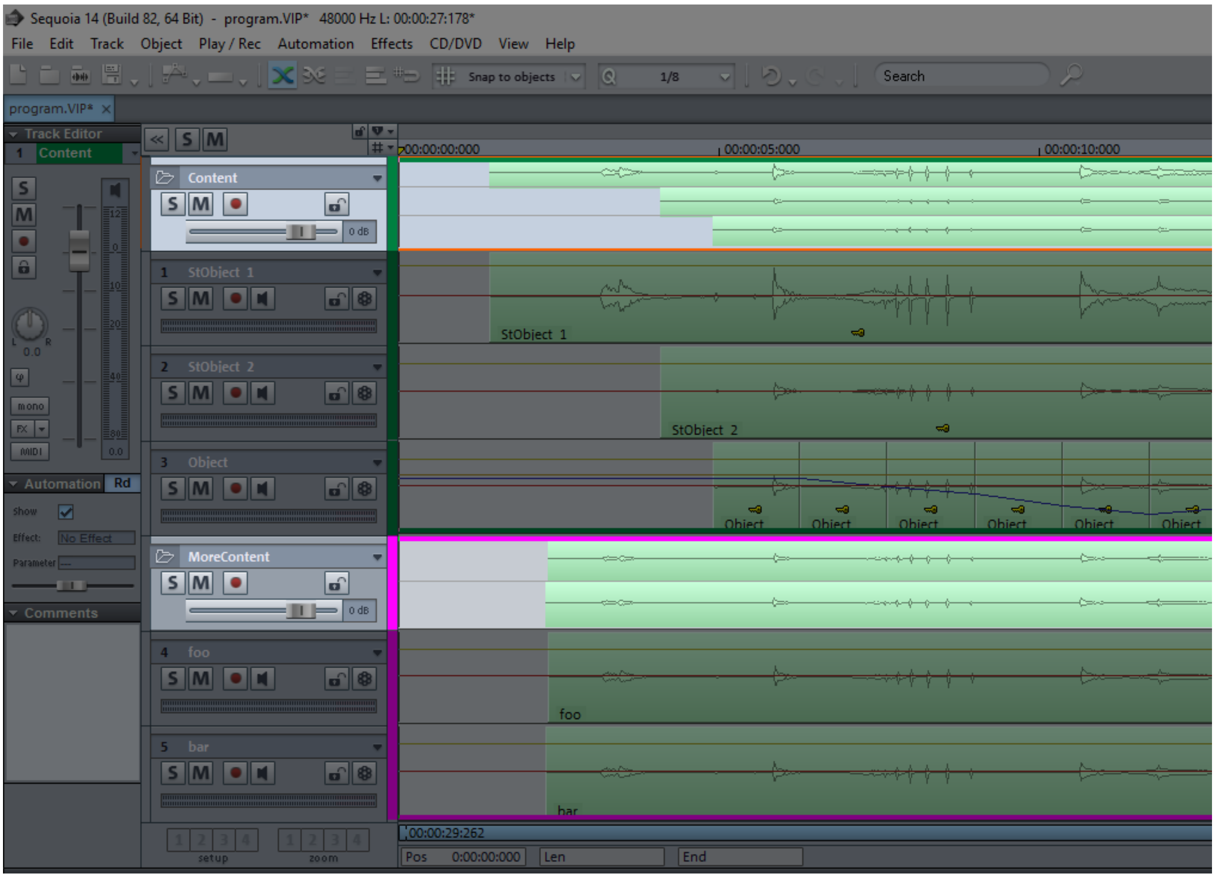


Figure 2: Highlighted folder tracks in Sequoia

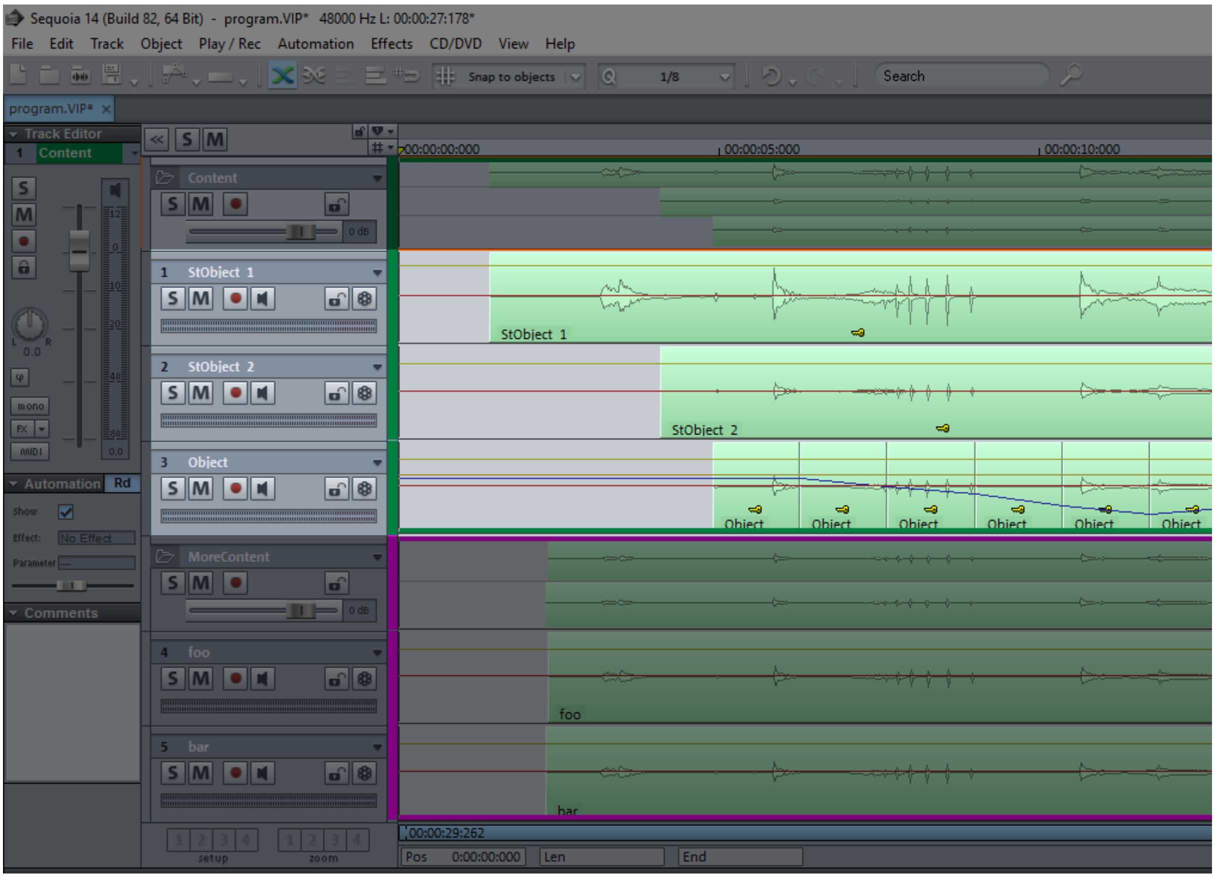


Figure 3: Highlighted audio tracks in Sequoia

The following rules and conventions are used:

- consecutive tracks can be grouped using name formatting (<objName>_1, <objName>_2, ...) to create a multichannel ADM object
- no nesting of folder tracks allowed
- empty tracks are not processed
- tracks outside of folder tracks are not processed
- audio block elements are created automatically according to automation curves and clip ranges

The following workflow can be used to directly record to ADM format:

- Create 32 channel recording project (VIP)
- In the mixer setup dialog:
    - choose the surround preset with the 32_0 preset
    - routing -> assign tracks to surround channels
- Add B-Com plugins on surround master
- Add a folder track with the name "##RECORDING"
- Add tracks to this folder to specify the target layout (e.g. 11 tracks for 7+4 setup)
- These tracks will be used to generate the ADM information
- Finally use File -> Save ADM File -> Audio export

ADM preview using MPEG-H renderer:

For previewing the audio and interpreting the ADM metadata the MPEG-H renderer from FH is implemented as an internal VST plug in.

In each Sequoia track an instance of the plug in collects the ADM metadata and sends them to the master instance, which is an insert in the surround master bus.

Here all meta data is processed and creates an audio signal including all ADM effects, e.g. panning to various surround setups such as 7.4.1, 7.1, 5.1 or 2.0.

Outlook:

Currently Sequoia can generate, read, write and edit ADM files containing audio clips, volume curves, pan curves (x,y, z) and name tags.

For pilot 1 (summer 2017) also interactivity features such as

- dynamic language selection

- foreground / background balancing and

- chapter skip / repeat

will be introduced.

Also a dedicated meta data editor will be introduced later to edit the ADM interactivity features in the project in a user friendly way.

2.9.5 Translation of ADM to MPEG-H

Metadata is an essential part of an object-based broadcasting system and within ORPHEUS there are two metadata-models - ADM and MPEG-H - for which a translation is needed. ADM is used in production and contribution, whereas MPEG-H is used in distribution to the end-user devices. Though the basic concepts of both metadata-models are similar in nature, the exact syntax and semantic can differ widely and one model may use specific elements which cannot be expressed in the other. One goal of ORPHEUS is therefore to define a common subset for which a direct translation of metadata between ADM and MPEG-H is possible.

Metadata in this context means audio-related metadata, describing characteristics of audio material (object-based, channel-based, and scene-based) as it is relevant for rendering and playback. It includes the position and gain of audio-objects as well as the channel setup of channel-based elements (e.g. 5.1+4H). In addition, metadata describing the structure and interactivity as relevant for the end-user experience need to be converted. While some of the above metadata is static, other may be dynamic, i.e. change over time (e.g. position of an object).

ADM is a very flexible and rather generic model, which uses XML as its primary representation language. In general, it allows an arbitrary amount of objects and can use a recursive hierarchy to structure those into groups. MPEG-H on the other hand, uses a relatively restricted bit-stream syntax, which is defined in MPEG-H Metadata Audio Elements (MAE) and other elements within the general configuration structure, named mpegh3daConfig(). In contrast to ADM, MPEG-H limits the hierarchy of audio elements to a single level. Further restrictions are imposed on the flexibility of MPEG-H when a specific *Profile and Level* is used. For example, the Low Complexity Profile at Level 3, as used in ATSC 3.0, DVB and ORPHEUS, restricts the number of simultaneously rendered objects to 16. Hence, it becomes obvious that the flexibility of ADM has to be limited when the distribution via MPEG-H is considered, as illustrated in Figure 4. This ongoing work will result in an ADM profile definition which is compatible to the MPEG-H LC Profile.

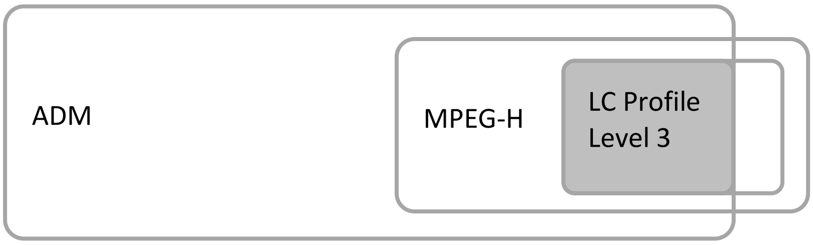


Figure 4: Illustration of the feature-space of audio-related metadata in ADM and MPEG-H. For a direct translation it is required to restrict the features to the common subset (shaded grey).

Some correspondences between ADM and MPEG-H metadata are listed in Table 7. The table only shows examples of *Structural Metadata* but similar correspondences also exist for metadata defining *Object Rendering* and *Interaction Control*. In addition, only the high-level syntax elements are listed and more detailed information on the attributes and semantic definitions are avoided for simplicity. The complete table includes close to 400 entries and builds the basis for the task of metadata translation.

Concept	**ADM Metadata**	**MPEG-H Metadata**
Signals and Formats	<audioChannelFormat>	Signals3d()
Speaker Layout	<audioPackFormat>	SpeakerConfig3d()
Grouping	<audioObject>, <audioContent>	mae_GroupDefiniton()
Exclusive-Or Selection	<audioComplementaryObjectIDRef>	mae_SwitchGroupDefinition()
Pre-Defined Mixes	<audioProgramme>	mae_GroupPresetDefinition()
…	…	…

Table 7: Correspondences between ADM and MPEG-H Structural Metadata (incomplete)

As indicated in Figure 4, the values of the metadata elements have to be constrained such that they are compatible with the Low Complexity Profile of MPEG-H at Level 3. Some of those constraints are listed in Table 8, which forms the basis for an ADM profile definition. It is possible to capture most constraints within an XML schema and an initial version of such a schema has been created based on the generic XML schema for ADM. Compliance with that modified ADM schema can be used as an indicator for an MPEG-H compliant ADM file.

MPEG-H Metadata (LC Profile, Level 3)	Restriction	Corresponding ADM Metadata
Nr. of groups	≤ 16	Nr. of audioObject elements Nr. of audioPackFormat elements Nr. of overall audioObjectIDRef references per audioContent Nr. of audioPackFormatIDRef references per audioObject Nr. of audioPackFormatIDRef references per audioStreamFormat
Nr. of presets	≤ 8	Nr. of audioProgramme elements
Nr. of preset conditions (per preset)	≤ 16	Nr. of audioContentIDRef per audioProgramme element
Nr. of switch groups	≤ 8	Nr. of audioObject elements using audioComplementaryObjectIDRef
mae_bsDescriptionDataLength	≤ 256	length of descriptors
sampling rate	∈ {48, 44.1, 32, 29.4, 24, 22.05, 16, 14.7} [kHz]	sampleRate attribute in audioTrackUID element
Nr. of loudspeakers in ref. layout	≤ 12	Restriction of specific audioPackFormatIDs for type="directSpeakers"
HOA order	≤ 6	order in audioBlock
(Nr. of objects without divergence) + 3·(number of objects with divergence > 0)	≤ 16	(Nr. of elements with objectDivergence == 0) + 3· (Nr. of objectDivergence elements with value > 0)
Nr. of objects with divergence and spread	=0	if width, height or depth are bigger than zero, then objectDivergence has to be zero width, height, depth of an object have to be zero if objectDivergence is bigger than zero
Temporal resolution of dynamic metadata for 1024 samples/frame	>256 [samples]	Duration of audioBlockFormat elements
…	…	…

Table 8: Restrictions derived from MPEG-H Low Complexity Profile, Level 3 (preliminary)

Though the metadata-models of ADM and MPEG-H are designed for a very high degree of compatibility, some caution is needed to translate from ADM to MPEG-H. In the following, we list some of the issues that need to be addressed in future work. Most can be resolved by restricting the ADM feature-space as outlined above.

- **Coordinate Systems:** ADM files with Cartesian coordinates are not directly translatable but need a coordinate transform.
- **Zone Exclusion / Excluded Sectors:** Resulting from the coordinate system transform, there might be minor incompatibilities with excluded sectors/zones, because ADM uses a cuboid

definition of zones and MPEG-H uses an angular definition.

- **Interpolation Length:** Interpolation length for dynamic metadata cannot be signalled in MPEG-H. The frame length should be aligned to simplify translation.
- **Signal Types:** Matrixed channels and pre-binauralized sound cannot be signalled explicitly in MPEG-H.
- **Speaker Distance:** The 'intended' loudspeaker distance (for the reference/target speaker positions of channel-based elements) cannot be signalled in MPEG-H
- **Screen Distance:** The Reference Screen Distance cannot be signalled in MPEG-H
- **Recursion:** There's no possibility to describe recursively nested groups in MPEG-H

Status of Implementation

Because MPEG-H metadata is defined as a binary format with bit-fields of various lengths, it is difficult to read and modify by humans and 3rd party software. Therefore, FHG has defined a corresponding XML-based format as an intermediate representation. This proprietary XML-format is also used to configure the software components from FHG such as the MPEG-H Encoder and the MPEG-H Production Library. Though this XML-format is still in development and subject to change, it is used as the basis for the work on metadata-translation. In other words, it is our initial goal to translate one XML-file containing MPEG-H metadata into another XML-file containing ADM metadata. The software framework for reading and writing both XML-formats into corresponding memory structures is available and work on the actual translation has started. As a reference for correct operation, some corresponding pairs of XML-files have been generated manually. Ultimately, this activity should result in a C-Code library which can be used in various components of the OPRPHEUS architecture.

2.10 Archiving

Archiving, in the sense of long-term storage of object-based audio content, is a crucial process within the broadcasting chain. Especially archives hosted by broadcasters may contain audio signals dating back decades, and provide content for all current and future distribution channels, but also serve as a basis for new productions. It is thus important that a format for archiving guarantees that content within the archive can be accessed and used without loss of information nor quality in the future.

As outlined on Deliverable 4.1, the formats for production and archiving should therefore fulfil certain requirements. The most important ones are

- The format should be publicly and openly documented.
- The format should be non-proprietary and the format should not depend on proprietary equipment.
- The format should not depend on a specific operating system or type of equipment.
- The format should be an uncompressed format.
- An archive file should include technical metadata
- The format should support sampling rates up to at least 48kHz, better up to 96kHz and a bit-depth of at least 24bit.

Given these requirements and based on an assessment of potential formats, the consortium has chosen the BW64 format as a container format (section 2.1), containing both ADM metadata (section 2.2) and PCM encoded, uncompressed audio samples (audio essence), to be used within the ORPHEUS project.

The BW64 format is specified in ITU-R BS.2088 [2], which is available publicly and free of charge. It allows to directly store the technical metadata along with the uncompressed audio data. As it is backwards compatible with the widely-used RIFF/WAVE format – if the file size does not exceed 4 Gb

– no special equipment is required to process the audio data within BW64 files. Finally, having native support to carry ADM metadata to represent object-based audio makes it a natural choice for adoption within the ORPHEUS project.

But, as one of the objectives of ORPHEUS is not only to provide a working end-to-end chain, but also investigate means to ease the adaption of object-based broadcasting for existing workflows, the consortium takes care of backward compatibility and interoperability.

Considering broadcasting archives, it is often not possible or feasible to start from scratch. Instead, new technologies must be integrated into an existing system with existing content. It is obvious that acceptance of an object-based audio workflow will be higher if the integration barrier with existing archives, both software and metadata, is as low as possible.

While investigations on this topic are still ongoing, at least two strategies can be identified. The first is to ensure convertibility and consistency between existing metadata models and the technical metadata required for object-based broadcasting. The work on compatibility with BMF as described in section 2.7 is one example of this approach. The second strategy is to consider BW64, the ADM or both as an interface specification for the archive. This approach has been taken with the import/export functionality of the stream-based storage of the IP studio as described in Deliverable 3.4 (“Implementation and documentation of a live object-based production environment”).

3 Provision Formats for Distribution to the End-user

For the distribution of object-based audio from the broadcaster to the end-user, the ORPHEUS project has selected two provision formats which are used in parallel. Both formats focus on the Internet as the most flexible transmission network and use **MPEG-DASH** as the underlying transport protocol.

Distribution to HTML5 Browsers: On the one hand, AAC and ADM are used to stream object-based audio to HTML5 browsers. This path allows distribution to a large audience as HTML5 browsers are widely available and content can therefore be consumed immediately. Because AAC per se is not an object-based audio codec, the additional functionality is implemented in Java Script and uses the WebAudio API for rendering in the browser.

Distribution to MPEG-H Clients: The second path is based on MPEG-H as a state of the art Next Generation Audio (NGA) codec. Because MPEG-H was designed from start with object-based audio in mind, it includes rendering and provides a well-defined audio quality and APIs, which also cover user interaction (UI). In addition, it is more bit-rate efficient and less complex than the browser-based solution. Within the ORPHEUS project, two clients are developed to verify MPEG-H as a universal delivery format for a diverse range of devices: An iOS-app running on an iPhone and a high-end AVR for the home.

In the following Section 3.1 we first provide some basic information on MPEG-DASH as this streaming format is used for both distribution paths in common. More detailed information on the specifics of the HTML5- and MPEG-H-based distribution path is then provided in Section 3.2 and 3.3 respectively.

3.1 MPEG-DASH

“Dynamic Adaptive Streaming over HTTP (DASH), also known as DASH, is an adaptive bitrate streaming technique that enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers. DASH is the first adaptive bitrate streaming solution that is an international standard. It was standardized by MPEG and is therefore called more precisely MPEG-DASH [7]. It already has significant market adoption and has a good chance of becoming the universal solution for media delivery on the Internet - outdating similar but more proprietary solutions like Smooth Streaming by Microsoft, or HDS by Adobe.

MPEG-DASH works by breaking the content into a sequence of small *Segments*, each segment containing a short interval of playback time, typically 2-10 seconds. As a container format for the raw audio bitstream the MPEG-4 File Format (MP4) [8] is used and each temporal segment then corresponds to an MP4 fragment (fMP4). In addition to the temporal segmentation, the content is made available at a variety of different *Representations*, i.e., alternative encodings of the same content at different bit rates. Based on the current network conditions, the DASH-client can then automatically select the appropriate Representation for download and playback. More specifically, the client selects the next segment with the highest possible bitrate that can be downloaded in time for play back without causing stalls or re-buffering events in the playback. Thus, an MPEG-DASH client can seamlessly adapt to changing network conditions, and provide high quality play back with fewer stalls or re-buffering events.

An essential element of the MPEG-DASH standard is the Media Presentation Description (MPD), which is an XML file describing metadata about the encoding and location of the Segments. It is need by the DASH-client to initialize the streaming session and is often referred to as the “index file” or “playlist”. Especially for the use case of a live service it is recommended to address Segments using template-generated URLs. Such a *SegmentTemplate* avoids the need to update the MPD in regular intervals and is also used for ORPHEUS.

MPEG-DASH uses HTTP/TCP as the transport protocol and can therefore use existing infrastructure which is widely available for content delivery over the World Wide Web. In particular, conventional HTTP severs can be used to stream media and Content Distribution Networks (CDN) can be used to scale the service to many users and geographical regions.

DASH is codec-agnostic which means it can stream content encoded with any codec like H.264, AAC or MPEG-H 3D Audio. However, each codec may need additional constraints and agreements in order to achieve true interoperability. Therefore the DASH Industry Forum (DASH-IF) is working on implementation guidelines and defines interoperability points for MPEG-DASH [9]. The guidelines and clarifications of DASH-IF shall also apply to ORPHEUS, in particular with respect to MPEG-H 3D Audio.

3.2 AAC + ADM Streaming

In order to deliver multi-object/channel audio to the browser in a form which is compatible with the majority of browsers and platforms we use an approach based on the MPEG-DASH standard. Audio and metadata are presented to the client as DASH segments, however unlike standard DASH play back in which a single audio representation is played back at a time we consume all of the audio adaptation sets simultaneously and schedule them for playback. This allows us to distribute higher channel counts than can be handled by the decoders bundled with popular browsers.

3.2.1 Encoding

Audio and metadata is captured in ORPHEUS using the IP Studio system, which handles hardware and network IO, timing and input alignment. Multiple audio inputs must be matrixed into groups of 5 channel streams. This must be done as only 5 channels groups can reliably decoded by most modern browsers. IP Studio can encode these streams in real time to multiple bandwidth AAC representations which are then passed to an IP Studio sequence store.

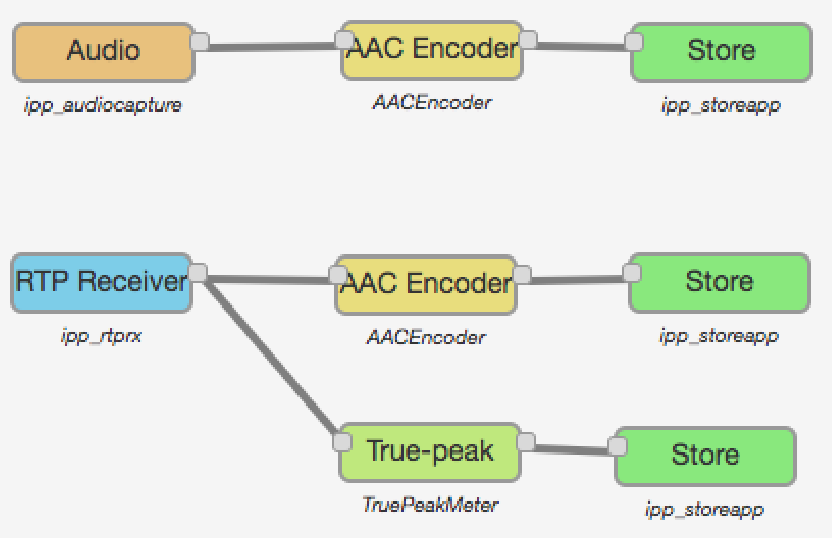
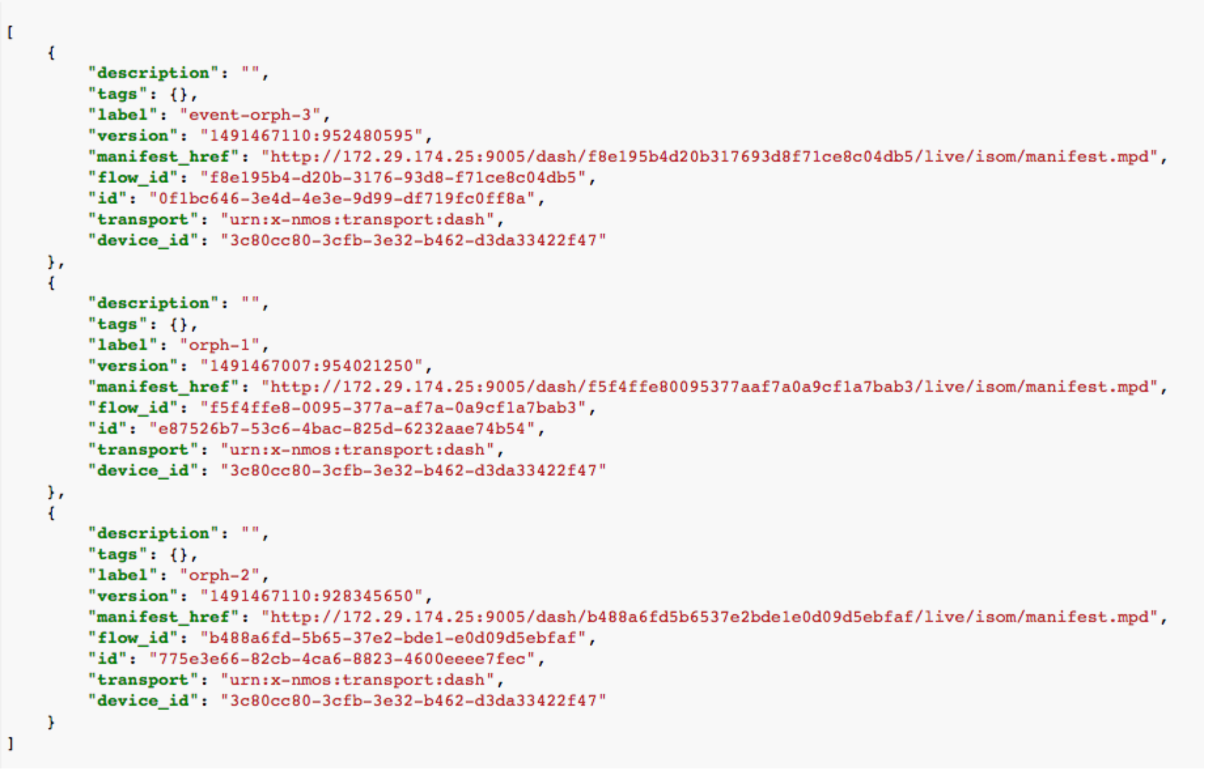


Figure 5: Simple IP Studio pipeline generating 2 stereo audio streams and a single metadata stream



```
[
    {
        "description": "",
        "tags": {},
        "label": "event-orph-3",
        "version": "1491467110:952480595",
        "manifest_href": "http://172.29.174.25:9005/dash/f8e195b4d20b317693d8f71ce8c04db5/live/isom/manifest.mpd",
        "flow_id": "f8e195b4-d20b-3176-93d8-f71ce8c04db5",
        "id": "0f1bc646-3e4d-4e3e-9d99-df719fc0ff8a",
        "transport": "urn:x-nmos:transport:dash",
        "device_id": "3c80cc80-3cfb-3e32-b462-d3da33422f47"
    },
    {
        "description": "",
        "tags": {},
        "label": "orph-1",
        "version": "1491467007:954021250",
        "manifest_href": "http://172.29.174.25:9005/dash/f5f4ffe80095377aaf7a0a9cf1a7bab3/live/isom/manifest.mpd",
        "flow_id": "f5f4ffe8-0095-377a-af7a-0a9cf1a7bab3",
        "id": "e87526b7-53c6-4bac-825d-6232aae74b54",
        "transport": "urn:x-nmos:transport:dash",
        "device_id": "3c80cc80-3cfb-3e32-b462-d3da33422f47"
    },
    {
        "description": "",
        "tags": {},
        "label": "orph-2",
        "version": "1491467110:928345650",
        "manifest_href": "http://172.29.174.25:9005/dash/b488a6fd5b6537e2bde1e0d09d5ebfaf/live/isom/manifest.mpd",
        "flow_id": "b488a6fd-5b65-37e2-bde1-e0d09d5ebfaf",
        "id": "775e3e66-82cb-4ca6-8823-4600eeee7fec",
        "transport": "urn:x-nmos:transport:dash",
        "device_id": "3c80cc80-3cfb-3e32-b462-d3da33422f47"
    }
]
```

Figure 6: NMOS Node API showing available live "senders" from which media can be consumed.

3.2.2 Segmentation

The sequence store generates a MPEG-DASH manifest and timestamped segments representing each 5 channel audio stream. These streams are time aligned so can be referenced independently.

Sequence stores also provide segmentation for events resulting in timestamped JSON segments representing the metadata to be applied to each segment of audio.

3.2.3 Transfer to public domain

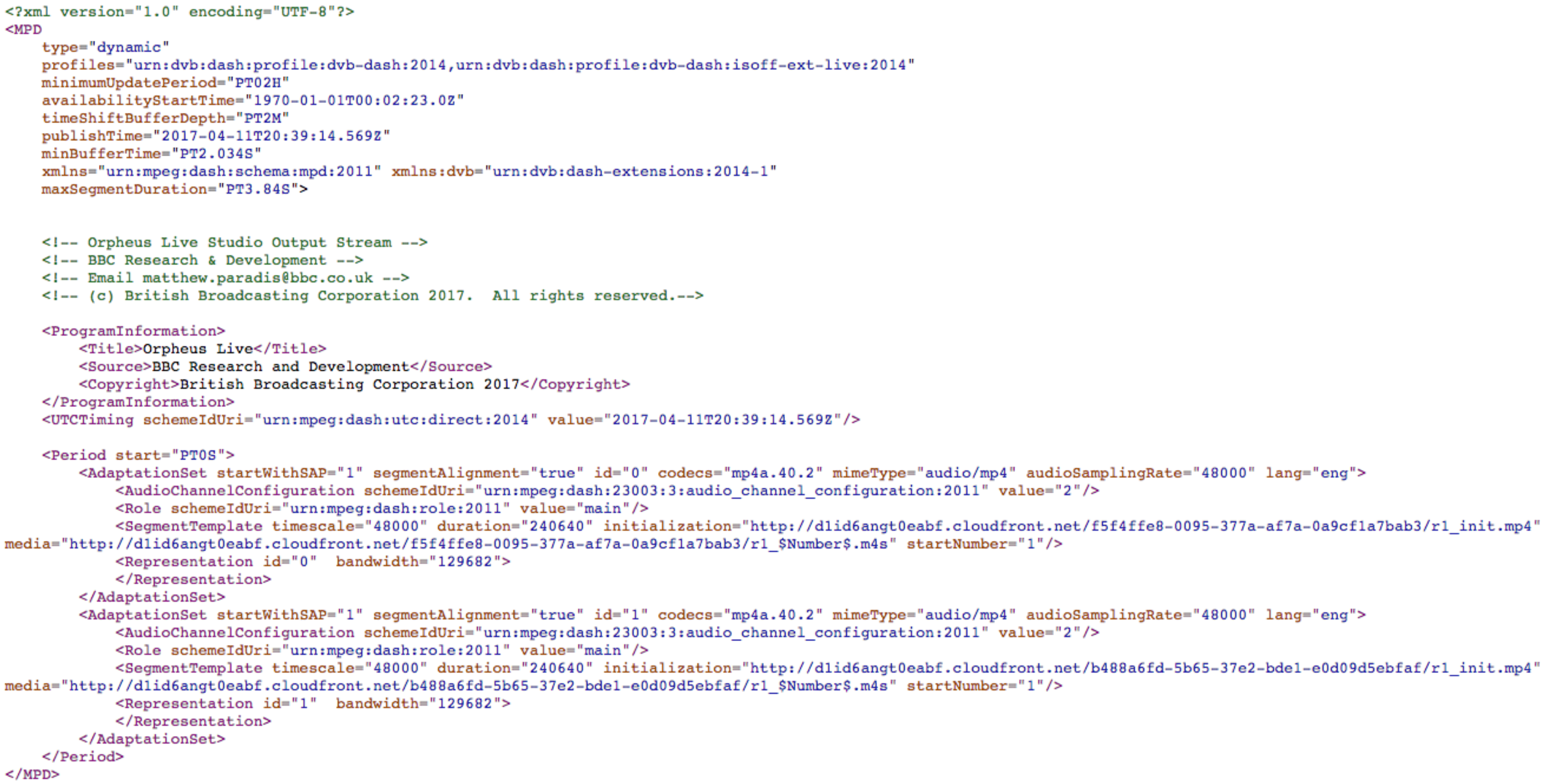
In order to transfer the media to the public audience for consumption, a process has been developed which consumes a collection of IP Studio manifests (1 per audio or metadata stream), and pushes segments to Amazon Web Services (AWS) S3 storage service.

Amazon CloudFront is used as a CDN to make these media and metadata segments available in an efficient manner globally to audiences. Any storage and CDN service can be used, however we chose to use Amazon as the BBC has an existing contract for those services.

3.2.4 Media reference

To allow the audience to access the media, an aggregation service is provided. When queried via a REST API this returns a manifest which contains separate adaptation sets for each media stream which makes up the broadcast along with adaptation sets for the metadata segments.

The service uses the timing provided on the ORPHEUS IP Studio node to provide a manifest which begins as close to the live edge as possible to provide users with a live listening experience as they would expect from a radio broadcast.



```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
    type="dynamic"
    profiles="urn:dvb:dash:profile:dvb-dash:2014,urn:dvb:dash:profile:dvb-dash:isoff-ext-live:2014"
    minimumUpdatePeriod="PT02H"
    availabilityStartTime="1970-01-01T00:02:23.0Z"
    timeShiftBufferDepth="PT2M"
    publishTime="2017-04-11T20:39:14.569Z"
    minBufferTime="PT2.034S"
    xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:dvb="urn:dvb:dash-extensions:2014-1"
    maxSegmentDuration="PT3.84S">


    <!-- Orpheus Live Studio Output Stream -->
    <!-- BBC Research & Development -->
    <!-- Email matthew.paradis@bbc.co.uk -->
    <!-- (c) British Broadcasting Corporation 2017.  All rights reserved.-->

    <ProgramInformation>
        <Title>Orpheus Live</Title>
        <Source>BBC Research and Development</Source>
        <Copyright>British Broadcasting Corporation 2017</Copyright>
    </ProgramInformation>
    <UTCTiming schemeIdUri="urn:mpeg:dash:utc:direct:2014" value="2017-04-11T20:39:14.569Z"/>

    <Period start="PT0S">
        <AdaptationSet startWithSAP="1" segmentAlignment="true" id="0" codecs="mp4a.40.2" mimeType="audio/mp4" audioSamplingRate="48000" lang="eng">
            <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <SegmentTemplate timescale="48000" duration="240640" initialization="http://d1id6angt0eabf.cloudfront.net/f5f4ffe8-0095-377a-af7a-0a9cf1a7bab3/r1_init.mp4"
media="http://d1id6angt0eabf.cloudfront.net/f5f4ffe8-0095-377a-af7a-0a9cf1a7bab3/r1_$Number$.m4s" startNumber="1"/>
            <Representation id="0"  bandwidth="129682">
            </Representation>
        </AdaptationSet>
        <AdaptationSet startWithSAP="1" segmentAlignment="true" id="1" codecs="mp4a.40.2" mimeType="audio/mp4" audioSamplingRate="48000" lang="eng">
            <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <SegmentTemplate timescale="48000" duration="240640" initialization="http://d1id6angt0eabf.cloudfront.net/b488a6fd-5b65-37e2-bde1-e0d09d5ebfaf/r1_init.mp4"
media="http://d1id6angt0eabf.cloudfront.net/b488a6fd-5b65-37e2-bde1-e0d09d5ebfaf/r1_$Number$.m4s" startNumber="1"/>
            <Representation id="1"  bandwidth="129682">
            </Representation>
        </AdaptationSet>
    </Period>
</MPD>
```

Figure 7: Aggregate Manifest generated from available media which is not located on AWS.

3.2.5 Playback and rendering

In order to playback the media referenced by the aggregation service we use a JavaScript library title bbcat-js (BBC Audio Toolkit – Javascript).

This library consumes the aggregated manifest and begins downloading segments into a buffer for all provided adaptations. These segments are decoded and queues as Web Audio API buffer sources for sample accurate playback. Transport controls such as pause, skip forward and backwards are also implemented.

In addition to the media playback any metadata segments that are available are handled in much the same way, however rather than being decoded as audio the JSON is parsed and rendering events are added to an internal queue. As the play-head progresses these events are sent to any number of available renderers in the library.

3.3 MPEG-H Streaming

The second distribution path is based on MPEG-H 3D Audio as a state of the art Next Generation Audio (NGA) codec with advanced capabilities with respect to immersion, personalization and universal delivery. For the distribution to the end-user two components have to be addressed: First, the MPEG-H 3D Audio codec as a generic solution for object-based 3D audio compression, second, the transport of MPEG-H over DASH as a universal solution for media delivery over the Internet.

3.3.1 MPEG-H 3D Audio

MPEG-H 3D Audio, specified as ISO/IEC 23008-3 (MPEG-H Part 3) [10], is an audio coding standard developed by the ISO/IEC Moving Picture Experts Group (MPEG) which was designed to meet the requirements of so called Next Generation Audio (NGA). It is adopted in broadcast application standards such as ATSC 3.0 and DVB and has been selected for terrestrial UHDTV services in Korea for which test streams are on air since April 2017.

The codec offers 3D sound to increase the realism and immersion of the user experience, and offers audio objects that enable interactivity or personalization by listeners. Immersive sound may be encoded using channel-based signals for typical loudspeaker setups or scene-based components encoded in Higher Order Ambisonics (HOA) in combination with static or dynamic audio objects. Interactivity can be enabled through broadcaster-authored preset mixes or through user control of object gains and positions. Improved loudness and Dynamic Range Control (DRC) allows tailoring the sound for best reproduction on a variety of consumer devices and listening environments.

The rendering of audio objects on arbitrary trajectories and the format conversion of loudspeaker configurations is an integral part of MPEG-H 3D Audio standard, which provides end-to-end control of the resulting audio quality. In addition, a binaural rendering module is included in the decoder, which can convey the spatial impression of immersive audio productions on headphones. This is of increased importance as media consumption is moving further towards mobile devices and personal audio with headphones as predominant form of playback.

The definition of audio metadata in MPEG-H 3D audio allows for personalized playback options, such as increasing or decreasing the level of dialog relative to the other audio content. With the metadata definition, MPEG-H 3D Audio also supports several use-cases for audio interactivity and object-based audio, such as changing the position of sound events, changing the language of a program, enabling of additional dialog tracks, choosing between content versions and automatic screen-related audio scene scaling. An overview of MPEG-H audio metadata is provided in [11]. For further information we also refer to [12] which is a comprehensive summary of the MPEG-H 3D Audio codec and its application in a TV broadcast environment.

Profile Level	**1**	**2**	**3**	**4**	**5**
Maximum sampling rate [kHz]	48	48	48	48	96
Maximum core codec channels in bit stream	10	18	32	56	56
Maximum simultaneously decoded core codec channels	5	9	16	28	28
Maximum loudspeaker outputs	2	8	12	24	24
Example loudspeaker configuration	2.0	7.1	7.1+4H	22.2	22.2
Maximum decoded objects	5	9	16	28	28

Table 9: Levels for low complexity profile of MPEG-H Audio [12]

The MPEG-H 3D Audio standard defines a complete NGA codec with many options and coding tools. In practice it is therefore required to define a subset for a specific application and complexity constraint. ORPHEUS follows the *Profile and Level* which has been defined in ATSC 3.0 and DVB, namely the Low Complexity Profile at Level 3 [13], see *Table* 9. Level 3 limits the codec to 32 core codec channels from which only 16 can be decoded simultaneously. For example, this allows decoding and rendering of 16 simultaneous audio objects or 3D speaker layouts such as 7.1+4H with 3 additional objects. Recommended core bitrates for the different channel configuration are listed in *Table 10*. The ATSC 3.0 application standard also includes further definitions and clarifications on the usage of MPEG-H, which shall also apply to ORPHEUS.

Channel Configuration	**Bitrate [kbs]**
2.0 Stereo	96
5.1 Multi-channel surround	192
7.1+4H Immersive Audio with 4 height speakers	384
22.2 Immersive audio	768

Table 10: Recommended core bitrates for excellent audio quality for broadcast applications [12]

3.3.2 MPEG-H over DASH

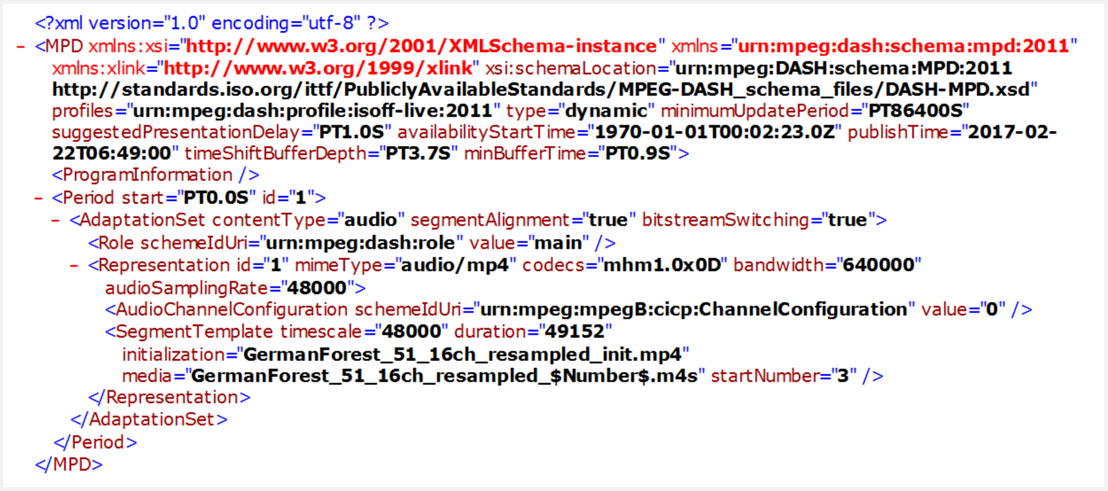
Because MPEG-H is primarily a compression format with the addition of 3D-rendering capability, it can be used like any other compression format together with codec-agnostic streaming formats such as DASH. MPEG-H is therefore defined as an optional audio codec in the DASH-IF Interoperability Guidelines since version 3.2 [9]. As ORPHEUS follows the MPEG-H profile and level definition from ATSC 3.0 it is consequent to also follow the Guidelines which ATSC and DASH-IF have defined jointly for the transport over DASH [14], in particular sections “5.4. Audio” and “5.4.4.3. MPEG-H Audio specific details” shall apply to ORPHEUS. In the following a high-level description of the relevant issues at the interface between MPEG-H and DASH is provided. For the detailed technical specification, we refer to the above mentioned sections.

All required metadata for decoding and rendering MPEG-H bit-streams is defined in the MPEG-H specification and is embedded in the MPEG-H Audio Stream (MHAS) in binary form. Therefore, the definition and encoding of all relevant metadata as well as its decoding and rendering behaviour is well defined when using MPEG-H as a codec format for streaming. However, the mapping between the metadata in production (e.g. ADM) and MPEG-H has to be assured.

In addition, MPEG-H offers the following features that are relevant for streaming: First, it supports seamless bit-rate switching through Instantaneous Playout Frames (IPF) which simplifies tune-in and adaptive streaming in DASH. MPEG-H also supports configuration changes and splicing on the bit-stream level, which can be important for use cases such as insertion of advertisements (“ad-insertion”). In addition, objects can be transmitted as independent streams and merged at the client, e.g. for hybrid delivery of an alternate language. Finally, the flexible rendering and format conversion of MPEG-H allows decoding the same bit-stream for e.g. a 5.1 surround speaker setup, stereo speaker setup, or binaural headphones. Hence, a single bit-stream is stored on the server covering multiple playback scenarios and therefore saving storage space and signalling complexity.

When initializing the streaming session it has to be considered that the overall audio scene may contain multiple channel beds and objects which can only be combined in certain ways. For example, an ambience atmosphere with 5.1 channels has to be combined with the dialog in either German or English language, and in addition an optional sound effect can be added. Though all those dependencies are well described in the MPEG-H Audio Scene Information (ASI) data structure, some of this information is also needed on the session initialization level, i.e. in the Media Presentation Description (MPD) of DASH. This allows, for example, that a DASH-client only fetches the English dialog track and therefore saves transmission bit rate. The mapping of audio scene information and dependencies onto the MPD has been defined in MPEG and DASH-IF under the term *Multi-Stream Delivery* and introduces the concept of *Bundles* and *PartialAdaptationSets* within the MPD syntax. Within ORPHEUS it was decided to start initially with a simple DASH scenario using a single MPEG-H stream and leave more advanced multi-stream scenarios for further study.

We conclude this section with an example MPD for the most basic use case for streaming MPEG-H 3D Audio over DASH in a live scenario. The MPD is shown in Figure 8 and corresponds to a single MPEG-H stream encoded at a single bitrate (Representation) of 640 kbit/s. The parameter *codecs=”mhm1.0x0D”* signals MPEG-H when using the Low Complexity Profile at Level 3. The *SegmentTemplate* is used to address the segments in the live stream as a template-generated URL. This MPD is part of the test data which is provided to ORPHEUS partners within the MPEG-H Decoder SDK.



```
  <?xml version="1.0" encoding="utf-8" ?>
- <MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:mpeg:dash:schema:mpd:2011"
    xmlns:xlink="http://www.w3.org/1999/xlink" xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011
    http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
    profiles="urn:mpeg:dash:profile:isoff-live:2011" type="dynamic" minimumUpdatePeriod="PT86400S"
    suggestedPresentationDelay="PT1.0S" availabilityStartTime="1970-01-01T00:02:23.0Z" publishTime="2017-02-
    22T06:49:00" timeShiftBufferDepth="PT3.7S" minBufferTime="PT0.9S">
    <ProgramInformation />
  - <Period start="PT0.0S" id="1">
    - <AdaptationSet contentType="audio" segmentAlignment="true" bitstreamSwitching="true">
        <Role schemeIdUri="urn:mpeg:dash:role" value="main" />
      - <Representation id="1" mimeType="audio/mp4" codecs="mhm1.0x0D" bandwidth="640000"
          audioSamplingRate="48000">
          <AudioChannelConfiguration schemeIdUri="urn:mpeg:mpegB:cicp:ChannelConfiguration" value="0" />
          <SegmentTemplate timescale="48000" duration="49152"
            initialization="GermanForest_51_16ch_resampled_init.mp4"
            media="GermanForest_51_16ch_resampled_$Number$.m4s" startNumber="3" />
        </Representation>
      </AdaptationSet>
    </Period>
  </MPD>
```

Figure 8: Example MPD for streaming MPEG-H over DASH

3.4 Implementation

Considering the distribution path to HTML5 browsers, the implementation approach and status is already documented in Section 3.2 and mainly under the responsibility of the BBC.

Considering the second distribution path based on the MPEG-H 3D Audio codec, Fraunhofer IIS has the role of providing the core software components and know-how while several other partners act as software integrators. More specifically, Fraunhofer IIS has provided three Software Development Kits (SDK) to ORPHEUS partners, namely

- MPEG-H Encoder SDK (BBC)
- MPEG-H Decoder SDK (BBC, ECANDY, TRINNOV)
- MPEG-H Production SDK (BBC, MAGIX, IRT)

While the scope of the Encoder- and Decoder-SDK should be obvious, the Production-SDK includes various tools for facilitating the production workflow. This includes the generation of MPEG-H metadata in the form of the *Control Track* (static and dynamic metadata encoded into a PCM audio channel using a modem) or an XML file (using a proprietary format aligned with the semantics of *Metadata Audio Elements* in MPEG-H, in particular mae_AudioSceneInfo()). In addition, the Production-Lib also includes 3D-rendering functionality and loudness measurement.

Each SDK consists of a core library for the intended target platform and additional material for supporting the integration process, including documentation, source code for example applications and test data. In addition, Fraunhofer IIS also developed and distributed additional "glue code" to facilitate or simplify the integration process. This includes an iOS-app which illustrates the User Interaction API of the MPEG-H decoder (relevant for ECANDY) and a DASH-receiver plugin based on the GStreamer media framework (relevant for TRINNOV).

Different end-user devices were developed to receive object-based audio content based on MPEG-H 3D Audio over MPEG-DASH. The details of these implementations are explained in D5.2 "Implementation and documentation of intermediate version of object-based renderers and user interfaces" and listed here for reference:

a) DASH client for MP4 streams with MPEG-H decoder in *Chromium* browser, by FHG (in chapter 3 of D5.2)

b) DASH client for MP4 streams with MPEG-H decoder in high-end AV receiver, by TRINNOV (in chapter 4 of D5.2)

c) DASH client for MP4 streams with MPEG-H decoder in iOS mobile application, by ECANDY (in chapter 5 of D5.2)

While the MPEG-H enabled Chromium browser was excluded as a reception device for the ORPHEUS pilots because of conflicts with the Open Source Software (OSS) licences which make a re-distribution to end users problematic, the integration for the other two reception devices is progressing well and initial versions are expected for the Pilot-1 in June 2017.

On the sender side, the BBC is planning to integrate the MPEG-H Encoder into its IP-Studio platform. Though this integration process has started, it is unclear whether it will be ready for deployment during the Pilot-1 timeframe. Therefore, ORPHEUS has decided to provide content for a "pseudo live" transmission as a backup plan. This content is pre-produced offline and pre-encoded with MPEG-H but distributed to the receiver devices "as if being live". Hence, the protocol interfaces (H over DASH) as well as the user experience is identical to a live trial and therefore suitable for testing and user evaluation.

4 Conclusions

This deliverable lists the formats selected for ORPHEUS and describes the interim status of the implementation of these formats for representation, archiving and provision of object-based audio. On the production side most of the formats are used, like BW64, ADM, NMOS and UMCP. For the provision or distribution MPEG-H and AAC + ADM metadata streamed over MPEG-DASH are used. The implementation and usage of the different formats in ORPHEUS and their capabilities are explained.

References

[1] ITU-R Recommendation BS.1352-3, File Format for the Exchange of Audio Programme Materials with Metadata on Information Technology Media. 2007, Intern. Telecom Union, Geneva, Switzerland. https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1352-3-200712-I!!PDF-E.pdf.

[2] ITU-R BS.2088-0, Long-form File Format for the International Exchange of Audio Programme Materials with Metadata. 2015, Intern. Telecom Union, Geneva, Switzerland. https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.2088-0-201510-I!!PDF-E.pdf.

[3] ITU-R BS.2076-0, Audio Definition Model. 2015, Intern. Telecom Union, Geneva, Switzerland. https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.2076-0-201506-I!!PDF-E.pdf.

[4] ITU-R BS.2094-0, Common Definitions for the Audio Definition Model. 2016, Intern. Telecom Union, Geneva, Switzerland. https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.2094-0-201604-I!!PDF-E.pdf.

[5] EBU Tech 3293, EBU Core Metadata set (EBUCore). 2016, European Broadcasting Union, Geneva, Switzerland. https://tech.ebu.ch/docs/tech/tech3293.pdf.

[6] EBU Tech 3351, EBU Class Conceptual Data Model (CCDM). 2016, European Broadcasting Union, Geneva, Switzerland. https://tech.ebu.ch/docs/tech/tech3351.pdf.

[7] ISO/IEC CD23009, Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats. 2014.

[8] ISO/IEC Int. Std. 14496-14:2010, Coding of Audio-Visual Objects - Part 14: The MP4 File Format. 2010.

[9] DASH Industry Forum, Guidelines for Implementation: DASH-IF Interoperability Points, Version 4.0. 2016. http://dashif.org/wp-content/uploads/2016/12/DASH-IF-IOP-v4.0-clean.pdf.

[10] MPEG-H ISO/IEC 23008-3 Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio. 2016, ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio.

[11] Füg, S., et al., Design, Coding and Processing of Metadata for Object-Based Interactive Audio, in 137th AES Convention. 2014: Los Angeles, USA.

[12] Bleidt, R.L., et al., Development of the MPEG-H TV Audio System for ATSC 3.0. IEEE Transactions on Broadcasting, 2017. 63(1), DOI: https://doi.org/10.1109/TBC.2017.2661258.

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

[14] DASH Industry Forum, Guidelines for Implementation: DASH-IF Interoperability Point for ATSC 3.0, Version 1.0. 2016. http://dashif.org/wp-content/uploads/2017/02/DASH-IF-IOP-for-ATSC3-0-v1.0.pdf.

Appendix A Example UMCP composition

The following example details a simple working UMCP Composition in UMCP v0.2. Two media sources are played and a mix processor switches between them.

In this example, a total of four Sequences are created; there are two Media Sequences that feed a ‘mixer’ Processor Sequence, which then feeds the Output Sequence. Note that UMCP is simply a collection of Events, and therefore a Sequence does not exist if there are no Events referencing it. To request the UMCP API to create a new Sequence, a ‘default’ Event must be created. This is acts as a placeholder for the Sequence by referencing the Sequence by ID. Additionally, it can provide initialisation instructions for the Sequence (such as default parameter values.)

A simple GET request to the API for a Composition will return a dump of the Composition as structured, hierarchical JSON. At the heart of this object are the UMCP Events that provide the instructions to produce the resultant media from the Composition. In this brief summary, some of the key elements of UMCP objects will be discussed.

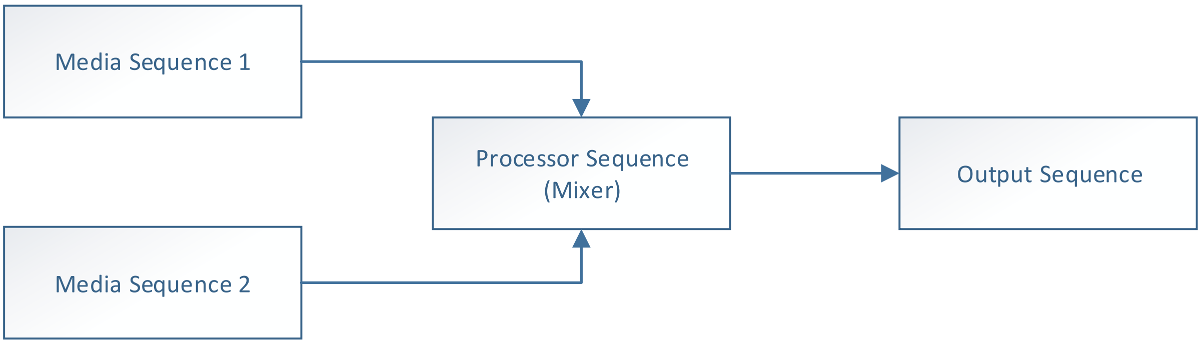
```
"999f381e-0389-4ea5-bd11-873254deaed2": {
  "title": "Audio Mix Composition",
  "id": "999f381e-0389-4ea5-bd11-873254deaed2",
  "sequences": {
    "674abcde-20d9-41cc-5573-000000000001": {
      "title": "Movie Trailer",
      "default_event": {...},
      "events": {...},
      "id": "674abcde-20d9-41cc-5573-000000000001"
    },
    "674abcde-20d9-41cc-5573-000000000002": {
      "title": "Big Buck Bunny",
      "default_event": {...},
      "events": {...},
      "id": "674abcde-20d9-41cc-5573-000000000002"
    },
    "674abcde-20d9-41cc-5573-00000000000a": {
      "title": "Audio Mixer",
      "default_event": {...},
      "events": {...},
      "id": "674abcde-20d9-41cc-5573-00000000000a"
    },
    "674abcde-20d9-41cc-5573-000000000000": {
      "title": "Audio Output",
      "default_event": {...},
      "events": {…},
      "id": "674abcde-20d9-41cc-5573-000000000000"
    }
  }
}
```

The top-most property in this structure provides the Composition ID. The value is an object containing a title property for the Composition, the ID, and a Sequences property. The Sequences property provides the IDs of all of the Sequences within the Composition, with a corresponding object. Within each Sequence object are the title and ID for the Sequence, and the Events within the sequence (including the aforementioned default event.) Note that the ‘Audio Output’ Sequence has no Events other than the default event. This is because Output Sequences do not perform any processing themselves. They simply signpost the output point of the processing pipelines. In some cases the Output Sequence may have Events, such as when connections are made to or from the Sequence over time. However, in this example, the connections are defined in the default event.

When defining a connection, the Event always occurs on the destination Sequence, not the source Sequence. Observing the default event of the Output Sequence, a connection is made from the Audio Mixer Sequence using its unique ID.

```
"default_event": {
 "event_payload": {
   "topic": "efe15658-fffa-4084-973a-5ddea35f1220",
   "type": "urn:x-ipstudio:format:event.composition.sequence.processors.audio.output",
   "data": [
    {
      "path": "input/0",
      "post": {
        "port": 0,
        "entity": "674abcde-20d9-41cc-5573-00000000000a"
      }
    }
   ]
 },
 "grain_type": "event",
 "origin_timestamp": "0:0",
 "source_id": "674abcde-20d9-41cc-5573-000000000000",
 "sync_timestamp": "1481040193:930000000",
 "creation_timestamp": "1481040193:930000000"
}
```

Similarly, the Audio Mixer Sequence connects from the two Media Sequences. It references input/0 and input/1 in the path properties to feed the Sequences in to separate mixer inputs. Ultimately, the following processing graph is produced.



The Event also includes some other important properties. The topic property provides a unique ID for this specific Event. The type property provides a URN describing the purpose of the Sequence. There is also an origin timestamp denoting the time at which this Event should be enacted on the Composition timeline. Since this is a default event, origin timestamp is irrelevant here. It should be noted however that all timestamps follow NMOS standards; that is to split them in to *second:nanosecond* pairs relative to the epoch.

The Audio Mixer Sequence provides a good example of how processor parameters are varied over time. The following data is take from the events property of the Sequence. The mix parameter is changed from 0 (all input 0) to 1 (all input 1) to 0.5 (mix of both).

```
"events": {
 "efe15658-fffa-4084-973a-5ddea35f1221": {
  "event_payload": {
    "topic": "efe15658-fffa-4084-973a-5ddea35f1221",
    "type": "urn:x-ipstudio:format:event.composition.sequence.processors.audio.mixer",
    "data": [
     {
       "path": "mix",
       "post": 0
     }
    ]
  },
  "grain_type": "event",
  "origin_timestamp": "1481000000:0",
  "source_id": "674abcde-20d9-41cc-5573-00000000000a",
  "sync_timestamp": "1481040193:930000000",
  "creation_timestamp": "1481040193:930000000"
 },
 "efe15658-fffa-4084-973a-5ddea35f1222": {
  "event_payload": {
   "topic": "efe15658-fffa-4084-973a-5ddea35f1222",
   "type": "urn:x-ipstudio:format:event.composition.sequence.processors.audio.mixer",
   "data": [
    {
     "path": "mix",
     "post": 1
    }
   ]
  },
  "grain_type": "event",
  "origin_timestamp": "1481000004:0",
  "source_id": "674abcde-20d9-41cc-5573-00000000000a",
  "sync_timestamp": "1481040193:930000000",
  "creation_timestamp": "1481040193:930000000"
 },
 "efe15658-fffa-4084-973a-5ddea35f1223": {
  "event_payload": {
    "topic": "efe15658-fffa-4084-973a-5ddea35f1223",
   "type": "urn:x-ipstudio:format:event.composition.sequence.processors.audio.mixer",
   "data": [
    {
      "path": "mix",
      "post": 0.5
    }
   ]
  },
  "grain_type": "event",
  "origin_timestamp": "1481000008:0",
  "source_id": "674abcde-20d9-41cc-5573-00000000000a",
  "sync_timestamp": "1481040193:930000000",
  "creation_timestamp": "1481040193:930000000"
 }
}
```

Finally, the Media Sequences provide an example of how Events are used to play media through a processing pipeline.

```
"events": {
 "efe15658-fffa-4084-973a-5ddea35f1241": {
  "event_payload": {
   "topic": "efe15658-fffa-4084-973a-5ddea35f1241",
   "type": "urn:x-ipstudio:format:event.composition.sequence.media.audio",
   "data": [
    {
     "path": "source_id",
     "post": "c39728ac-642f-418b-80dc-72762b2eceac"
    },
    {
     "path": "start_time",
     "post": "978307231:957333333"
    }
   ]
  },
  "grain_type": "event",
  "origin_timestamp": "1481000000:0",
  "source_id": "674abcde-20d9-41cc-5573-000000000002",
    "sync_timestamp": "1481040193:930000000",
    "creation_timestamp": "1481040193:930000000"
  },
  "efe15658-fffa-4084-973a-5ddea35f1242": {
    "event_payload": {
      "topic": "efe15658-fffa-4084-973a-5ddea35f1242",
      "type": "urn:x-ipstudio:format:event.composition.sequence.media.audio",
      "data": [
        {
          "path": "source_id",
          "post": null
        }
      ]
    },
    "grain_type": "event",
    "origin_timestamp": "1481000015:0",
    "source_id": "674abcde-20d9-41cc-5573-000000000002",
    "sync_timestamp": "1481040193:930000000",
    "creation_timestamp": "1481040193:930000000"
  }
}
```

These Events instruct the media source with ID "c39728ac-642f-418b-80dc-72762b2eceac" to playback from timestamp "978307231:957333333". This media will be accessible via the IP Studio Media Access API. It could be an audio file, or it may in fact be the output of another UMCP Composition. Fifteen seconds later, playback is stopped by setting the source ID to null.