Deliverable

Project Acronym:	**IMAC**
Grant Agreement number:	761974
Project Title:	*Immersive Accessibility*

D3.1. Architecture Design

Revision: 0.4 (Draft 1)

Authors: Chris Hughes (USAL), Peter tho Pesch (IRT), Mario Montagud (i2CAT), Marc Brelot (MSE), Romain Bouqueau (MSE) and Enric Torres (ANG)

Delivery date: M06 (31-03-2018) M16 (31-01-2019)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 761974

Dissemination Level		
P	Public	P
C	Confidential, only for members of the consortium and the Commission Services	

Abstract:

This document describes the architecture design and considerations of the ImAc project. It does this by providing a blueprint describing what we need to do, based on the user requirements gathered in D2.3. It also explains how we are going to do it, which we are going to use to make the platform work and outlines a series of metrics for testing to prove the system is technically successful.

REVISION HISTORY

Revision	Date	Author	Organisation	Description
0.1	01-12-2017	Chris Hughes	USAL	Template and ToC
0.2	05-03-2018	Chris Hughes	USAL	Draft Allocation of Sections
0.3	28-03-2018	Chris Hughes	USAL	Towards First Draft
0.4	15-04-2018	Chris Hughes	USAL	First Draft
1.0	30-04-2018	Chris Hughes	USAL	After Review

EXECUTIVE SUMMARY

This document is presented in two iterations and describes the technical architecture of the ImAc project. As the first iteration it is a working document providing a blueprint for the technical implementation of the project, and will be updated after the initial prototype implementation and testing phase to reflect changes needed to improve the platform.

The technical architecture is directly based of the user requirements established in WP2, specifically the platform specification (D2.3) that identifies the essential system requirements needed to ensure a successful implementation. In turn it feeds directly into the Integration and Testing Report (D3.6), which is designed to evaluate the success of each phase of implementation.

Chapter 1 provides an overview of this document, describes the objectives and scope of the technical architecture, and details how it fits into the larger ImAc project.

Chapter 2 describes the Goals and Constraints of the system architecture. These are the key requirements, which have a significant bearing on the architecture and they link directly to the end user requirements defined in the requirements table in D2.2. Satisfying these needs is essential to developing a successful framework for the ImAc project.

Chapter 3 provides an architectural representation of the project and establishes the architecture in terms of usage scenarios, logical and process views and a deployment structure.

Chapter 4 describes external dependencies, which will be used within the implementation of the project, where they are used within the architecture, and how they will be integrated with.

Chapter 5 discusses the size and performance requirements of the implementation. This details the key sizing and timing requirements for the platform to be used successfully, as stipulated in the Platform Specification, which will provide the basis for the testing strategy.

Chapter 6 concludes the document with a summary of the technical architecture.

Closed and open pilots in WP5 will allow testing and validation of the results from WP3 & WP4 and further refine this architecture in a second iteration of the ImAc platform specification.

CONTRIBUTORS

First Name	Last Name	Company	e-Mail
Chris	Hughes	USAL	c.j.hughes@salford.ac.uk
Mario	Montagud	i2CAT	mario.montagud@i2cat.net
Marc Romain	Brelot Bouqueau	MSE	marc.brelot@gpac-licensing.com
Enric	Torres	ANG	enric@anglatecnic.com
Peter	tho Pesch	IRT	thopesch@irt.de

CONTENTS

TABLES OF FIGURES AND TABLES

LIST OF ACRONYMS

Acronym	Description
AD	Audio Description
AST	Audio Subtitles
ST	Subtitles
SL	Sign Language
HUR	Home User Requirement
PUR	Professional User Requirement
HMD	Head Mounted Display
FOV	Field of View
CDN	Content Delivery Network
ISP	Internet Service Provider

1. INTRODUCTION

1.1. Purpose of this document

The final goal of WP3 is to define and implement a platform integrating different components of the production chain, including accessible content management, packaging and distribution, customisation of the experience, and display of immersive and adapted content. The design will be fed by requirements gathered in WP2 (T2.2. and T2.3). This includes:

- To design the architecture of a robust platform capable to integrate all the components developed in the project.
- To design and implement a content management component facilitating access to multiple content formats and its storage.
- To adapt and integrate a real-time process to effectively encode multiple streams from inclusive content (i.e. subtitles, audio description and sign language) into full omnidirectional video.
- To design and implement a delivery chain that can process the input from Production (especially for the Accessibility and Immersive sides) and make them available accurately to the player using standard technologies.
- To design and implement a player (including clients and libraries) required to display omnidirectional video across devices (TV, second screen and HMD) maintaining coherence, synchronization, and enabling interaction and personalisation features.
- Validate development in semi-open pilots and large-scale pilots.
- Disseminate and communicate the WP outcomes among other researchers and industry stakeholders.

This document provides the technical architecture, which describes the blueprint for each of the subsequent stages of WP3.

1.2. Scope of this document

This document will provide a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions, which have been made on the system based on User requirements (D2.2), and Platform Specifications (D2.3) defined in WP2.

The document follows a logical progression consisting of four main sections:

- Chapter 2: Architectural Goals and Constraints
- Chapter 3: Architectural Representation
- Chapter 4: Dependencies
- Chapter 5: Size and Performance

Chapter 2 describes what we need to do. This is based on the user requirements gathered in D2.3. Chapter 3 explains how we are going to do it. This will contain the blueprint for the ImAc system. Chapter 4 details the tools we are going to use to make the platform work. Finally Chapter 5 discusses the size and performance requirements of the implementation of each component of the ImAc system.

1.3. Status of this document

This is the first version of D3.1 with delivery foreseen in M06. A revised version of this document will be delivered in M16.

1.4. Relation with other ImAc activities

The PERT diagram illustrates the relation between D3.1 and the other ImAc activities. The Architecture design is built based upon the findings of D2.3 Platform Specification and it feeds into the implementation of both the Accessibility Services and the Immersive Platform.

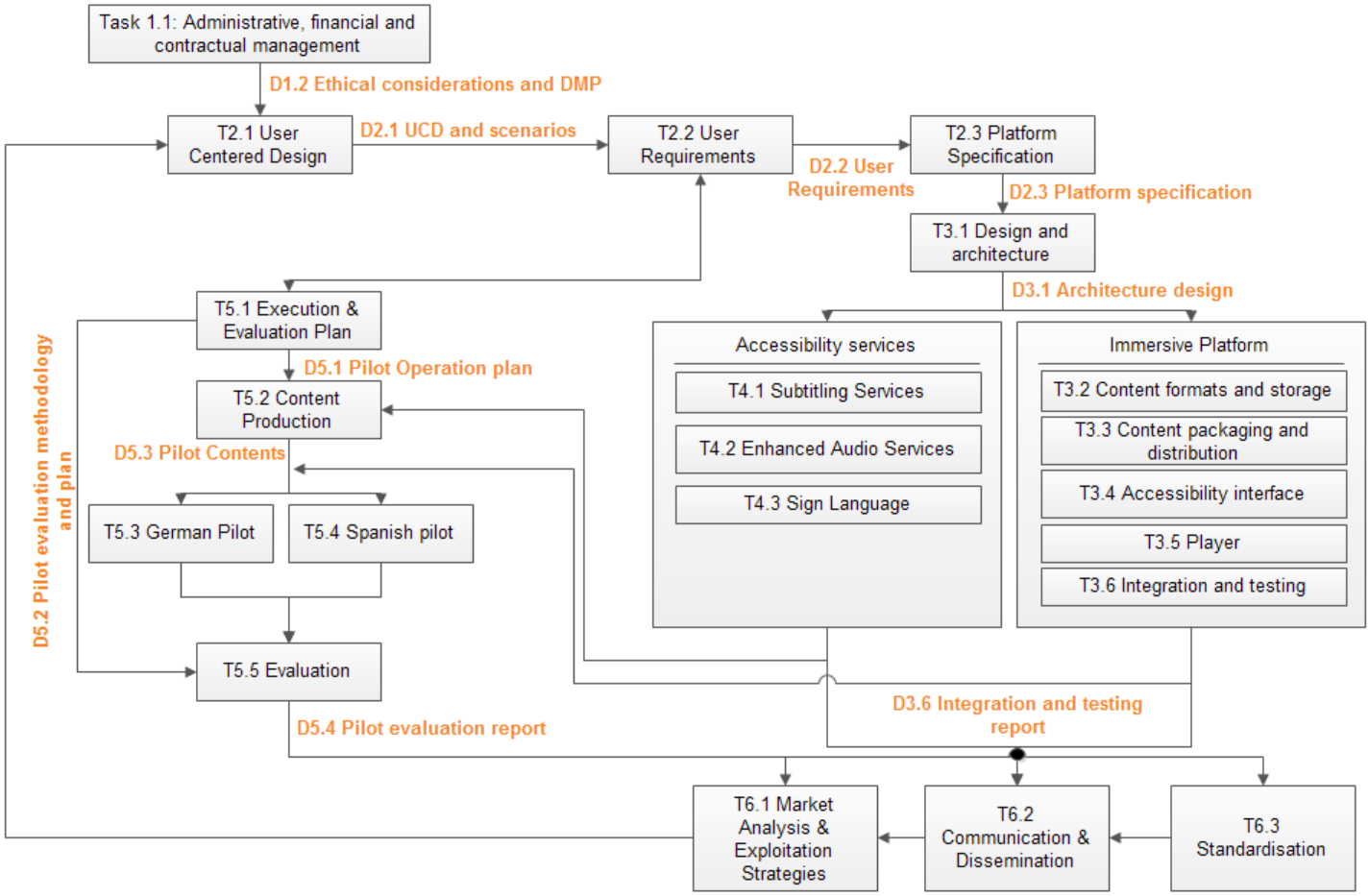

Figure 1 - PERT Diagram illustrating the relationship between D3.1 and other ImAc activities.

2. Architectural Goals and Constraints

2.1. Key Requirements

D2.2 followed a user centric design approach to understand the needs of users and provided a detailed description of the user requirements for the ImAc platform. The key requirements and system constraints, which have a significant bearing on the architecture, are listed in Table 1. These link directly to the end user requirements defined in the requirements table in D2.2 and satisfying these needs is essential to developing a successful framework for the ImAc project.

Table 1 - Key requirements taken from D2.2 that are required for a successful implementation.

Req. Nr.	Title
HUR.02.01.0	Access to subtitles
HUR.02.02.0	Access to signer
HUR.02.03.0	Access to Audio Description
HUR.02.05.0	Access to Audio Subtitling
HUR.02.06.0	Multiplatform player for desktop, mobile phone (cardboard supported, gyroscope sensor based), TV
HUR.02.08.0	Subtitles always on Main Screen
HUR.02.09.0	Playback of audio description
HUR.02.16.0	Switch on/off signer
HUR.02.17.0	Selection of personalisation options for signer
HUR.03.01.0	Sign Language Service
HUR.02.18.0	Accessibility interface signer - basic presentation mode
HUR.02.19.0	Accessibility interface signer - position in viewing field
HUR.02.24.0	Switch on/off subtitles
HUR.02.25.0	Selection of Personalisation options for subtitles
HUR.02.26.0	Accessibility interface for subtitles - basic presentation mode
HUR.02.27.0	Accessibility interface for subtitles - position in viewing field
HUR.02.32.0	Customization of immersive subtitle

HUR.03.03.0	Accessibility interface for subtitles - notices for dramaturgically significant sounds
HUR.02.34.0	Accessibility interface for subtitles - angular-based positioning mechanisms for subtitles placement
HUR.02.35.0	Accessibility interface for subtitles - adaptable subtitles speed
HUR.03.05.0	Immersive Subtitles information
HUR.02.36.0	Switch on/off audio description and audio subtitles
HUR.02.37.0	Selection of Personalisation options for audio description and audio subtitles
HUR.03.06.0	Playback of 3D audio
HUR.02.38.0	Accessibility interface for Audio Description -Persistent soundscape
HUR.02.42.0	Accessibility interface for Audio Description - mapped to event
PUR.01.01.0	Watch low-res preview content
PUR.01.02.0	Watch hi-res preview content
PUR.01.03.0	Navigate preview content by angle
PUR.01.04.0	Navigate preview content by frame
PUR.01.05.0	Navigate preview content by time
PUR.01.06.0	Navigate preview content by audio
PUR.01.07.0	File operations
PUR.01.08.0	Add subtitle text
PUR.01.09.0	Add sign language video
PUR.01.10.0	Create AD preview content
PUR.01.11.0	Add AD preview audio to video
PUR.01.12.0	Preview video and AD audio
PUR.01.13.0	Add audio description
PUR.03.01.0	Accessing content for ImAc enrichment

PUR.03.04.0	Triggering content packaging and distribution
PUR.03.06.0	Locally retrieving the packaging result
PUR.03.07.0	Directing the packaging result to a different resource

2.2. Considerations for an end to end subtitle workflow

This chapter describes the overall approach for ImAc's subtitle service from a technical perspective. Based on the initial user requirements that were collected from the focus group tests, this strategy was drafted to provide a technical basis for the service that considers all aspects of the subtitle workflow. Since modifications of the subtitle service can be expected during upcoming tests and pilot phases, the system provides flexibility especially regarding changes of subtitle presentation options.

The chapter is divided into two sections:

- A general part, which includes specifications that have implications on the overall system
- A content related part, which mainly describes considerations for the different requirements on the behaviour and presentation of subtitles

2.2.1 General specifications

Table 2 – List of relevant links regarding subtitle service

Description	Link
IMSC	https://www.w3.org/TR/ttml-imsc1.0.1/
TTML1	https://www.w3.org/TR/ttml1/
TTML2	https://www.w3.org/TR/2018/CR-ttml2-20180313/
EBU-TT	https://tech.ebu.ch/subtitling
WebVTT	https://www.w3.org/TR/webvtt1/
EBU STL	https://tech.ebu.ch/docs/tech/tech3264.pdf
DVB BlueBook A174	https://www.dvb.org/resources/public/standards/a174_dvb_ttml_subtitling_systems.pdf
HbbTV 2.0	http://www.hbbtv.org/resource-library/#specifications
Overview of various TTML profiles by W3C	https://www.w3.org/AudioVideo/TT/docs/TTML-Profiles.html
Imsc.js	https://github.com/sandflow/imscJS

2.2.1.1 Subtitle Distribution Format

ImAc's subtitle format for distribution is a subset of the TTML (Timed Text Mark-up Language). The TTML profile chosen in ImAc is IMSC text profile (Internet Media Subtitles and Captions) text profile.

The only relevant alternative to TTML format(s) is WebVTT (Web Video Text Tracks Format), a W3C working draft that specifies style, position and time mark-up for HTML text tracks. This document does not claim to provide a detailed comparison between TTML and IMSC. In ImAc TTML was chosen, mainly because:

- TTML and especially some subsets (EBU-TT, SDP-US) were specified to fit for professional use (e.g. by broadcasters). WebVTT on the other side is web related and not suitable for other areas of the subtitle workflow.
- TTML serves all areas of subtitle workflows, from production to distribution. It includes mark-up information and metadata to feed all distribution
- There is a very active community pushing TTML (especially some subsets). This includes stakeholders from the industry, content providers and standardization groups.

However, it has to be stated that WebVTT is widely supported by browsers today already. TTML (and IMSC) still lacks native browser support.

Various profiles and subsets of TTML exist, some are closely related and only differ in a few aspects. Other TTML subsets are for example: EBU-TT, EBU-TT-D, EBU-TT Part 3 (Live), SDP-US, CFF-TT, SMPTE-TT.

IMSC was chosen for various reasons:

- IMSC is the TTML profile that draws the most attention by standardization bodies and the industry right now.
- It can be extended with additional metadata and extensions.
- It is compliant with the current Netflix requirements.
- In 2017, the DVB referenced IMSC for their TTML Subtitling Systems (BlueBook A174).
- Since it is (almost) a superset of EBU-TT-D, an IMSC decoder will play EBU-TT-D streams. EBU-TT-D is referenced in the DASH profile for HbbTV2.0 for example.
- It's likely that in near/mid-term future, there will be a wider support for IMSC in the market.
- Most interesting for us (in IMAC) is probably that the JS library for subtitle rendering is an IMSC renderer.

Note: In the following, both formats TTML and IMSC are referred to. When talking about general aspects, that are the same for TTML and all its subsets, then TTML is used. When talking about ImAc's subtitle format, IMSC is used. However, there is no hard line in this rule and usage may not always be consistent.

One aspect of the subtitle service is streaming capabilities. Basic requirements in ImAc are similar to traditional subtitle streaming, but shall be mentioned here nevertheless:

- It must be possible to stream subtitles within common streaming formats.
- It must be possible to stream and identify multiple subtitle streams with different languages for one program.

The IMSC format can be streamed within the MPEG-DASH streaming format, which is used in ImAc – see chapter 3. As basis for the implementation, ImAc refers to the specification for transportation of EBU-TT-D in HbbTV 2.0. In HbbTV 2.0, two options are described: in-band

delivery and out of band delivery. Both are options in ImAc, although the out of band solution might be the simplest way to realize first implementations. In the framework of ImAc, multiple subtitles might be also transmitted for one program into the MPEG-DASH stream(s). Synchronization capabilities of multiple subtitles will be proposed (and will be signalized accordingly).

2.2.1.2 Subtitle Production Format

It is important for the ImAc architecture to specify the subtitle distribution format. And as indicated above, the IMSC format was chosen to support an easy integration into production workflows. Nevertheless, other formats WILL be used during the production process, which would then be transformed to IMSC for play-out. Production and distribution use different formats although TTML attempts to unify the technology by providing a production alternative of distribution EBU-TT-D as EBU-TT. This format is based on Teletext and the production format in most broadcaster environments. That includes, that STL is supported in all professional subtitle editor tools.

In ImAc, subtitle editing will be done using IMSC directly, but the EBU STL format will be supported as an import format, to ensure, that traditionally produced subtitle files can be used for the ImAc services.

2.2.1.3 Management of subtitle files

The management of subtitle files within the premises of a service provider is an important task in order to provide a reliable subtitle service (to professional and end users). But the workflows of different providers vary and depend very much on the overall system environment of a company. Thus, it is not in the scope of ImAc to draft a generic architecture for data handling.

In most cases, such workflows are independent of the content of a file, except for some metadata that ensures correct processing of the files. Example: The information of the subtitle language is required for a correct signalization of the subtitle stream.

However, ImAc installs an exemplary workflow for accessibility data. The Content Manager for accessibility data demonstrates how management of accessibility data can be smoothly included in production and distribution workflows. More details can be found in Chapter 3.

2.2.2 Content related considerations

2.2.2.1 Positioning of subtitles

Following the collected user requirements, a new extension for subtitle formatting will need to be added in ImAc: spatial information. In traditional (2D) media, subtitles are positioned as well, but for the concepts that are envisioned in ImAc, positioning plays a completely different role. Some essential differences arise when the video is mapped to a projection plane and only a part of the video is rendered: In 360°, subtitles and video are not linked as they are with "normal" video. The subtitle will never be distorted, it will not change size due to zooming, it will never be out of the Field of View (FOV) and a subtitle will not look like it is part of the scene.

The final rendering position of a subtitle may be calculated only on the user device where the information on the current viewing direction is available. That means the player needs to implement additional processing for correct subtitle rendering.

For the following considerations, some characteristics of a 360° environment are defined (these definitions comply with the ImAc system):

- The video is stored as a 2D image that is intended for an equirectangular projection.
- For presentation, this 2D image is mapped (projected) to a sphere. Coordinates on the sphere can be expressed by latitude and longitude.
- The viewer will see a part of this sphere that is projected (back) to a flat screen. This is called “field of view” (FOV).
- Within the FOV, the subtitles are placed within a safe area. That means the subtitles will never be placed at the very edge of the field of view (as it is done in traditional TV as well).

Typically, the subtitle-rendering plane (or “root container” in TTML) is equal to the full width and height of the underlying video. But it is not possible to simply follow that practice in ImAc, since the FOV (where subtitles shall be rendered on) is a) smaller than the video size; and b) not at a fixed (video) position. The TTML root container for the 360° environment could be defined in two different ways:

1. The rendering plane is defined as the FOV.
2. The rendering plane is defined as the stored 2D video, following the same equirectangular projection mapping (i.e. a location on the 2D rendering plane can be mapped to a point on the sphere).

In each approach the subtitles have to be processed for final rendering. Processing regarding icons that indicate directions (arrows, wind rose, etc.) will be similar for both options, only the information for speaker position may come from different attributes.

For ImAc’s first pilot phase, option 1 has been chosen. The advantage of option 1 is that the TTML document could be rendered onto the FOV without processing (of course any information regarding speakers position would be missing in that case). The document would be similar to a file for 2D media content and would also be backward compatible.

Speaker positions will be added by user-defined extensions, which is allowed by TTML and IMSC (as foreign namespace extensions). Introducing these extensions into TTML is the preferred way in ImAc in order to push standardized solutions. The speaker position information will be used by the player to add icons (arrows, wind rose) that point to the speaker or to re-position the subtitles in the FOV in relation to the current speaker position.

2.2.2.2 User preferences

This section will not repeat what user preferences will be available in ImAc (D2.3) but describe how they might be realized in the system. Three scenarios are possible:

- Transmit user preferences with dedicated extensions (e.g. font sizes in the subtitle format). The player makes preferences available to the user in a way that is described in the content stream.
- Handle user preferences in the player. In such a case, the content stream may include no metadata regarding preferences.
- Mixed approach: Some user preferences are indicated in the stream (e.g. what presentation modes should be offered), others are handled in the player only (e.g. font sizes).

It is a matter of sharing rendering responsibility between player application and content provider, and there will be no sharp line here, because a player may overwrite preferences of the content stream. For some preferences, it makes sense to have at least an "editor's choice" signalled within the stream.

In the first pilot phase, all user preferences will be realized within the player application, because it is the easiest way to realize early pilots in ImAc. In the second phase, it might be considered to move some of the information for user preferences to the content stream.

TTML can easily be enriched with user extensions, thus the subtitle format will not be a limiting factor. However, finding a standardized way for example for indicating available font sizes is preferred in ImAc.

2.2.2.3 Depth

The focus of ImAc is on environments with all elements of a scene (like media content and user interface elements) having the same depth. However, real 3D environments shall also be addressed in the project. For subtitles a few issues need to be investigated here, but this document will not provide a strategy for presentation of subtitles in 3D environments.

User requirements in this field were not collected yet in the project. Still, a basic assumption can be made: That subtitles should be rendered on top of (all) other objects, such that the text is not obscured by anything. The background for this assumption: ImAc's focus group questionnaires have revealed that users prefer subtitles to be always visible in the FOV. Apparently, users want to able to read them any time, which is comprehensible. The same demand can be assumed for 3D, which means the subtitle text must be visible at all time. This assumption leads to the following question:

How can subtitles be presented in a comfortable way, when the scene contains elements close to the viewer?

When subtitles are placed as another element into the scene and when a comfortable depth for that element is chosen for that subtitle, that means that no element can be placed closer to the viewer than the subtitle is (at least not at the same location). Adding subtitles in a real 3D environment will probably demand for compromises one way or another.

TTML supports basic options for specifying depth information on subtitles. The options available are likely to be sufficient to indicate at what depth the subtitle should be rendered.

2.2.2.4 Safe area

The size of the safe area when displaying subtitles on head mounted devices (HMD) is being evaluated within the pre-pilot tests. A concrete value cannot be defined yet and may vary

depending on the display used (on HMD smaller save areas might be more comfortable). For television systems the EBU R95 defines a graphic save area of 90% of the active picture.

In TTML, the save area is typically handled by the size (and position) of the region element. In 360°, the save area could be indicated the same way, but only when the TTML root container is defined to have the same size as the FOV (which will be the case in IMAC). An alternative option is that the player handles the save area by addressing a corresponding rendering plane. This plane should then be fully used (without leaving margins, because the player already considered safe area). Here, the editor would have no possibility to indicate the save area (at least not with native TTML features).

For the first pilot phase, ImAc will handle save area only in the player. Region sizes from the TTML document will be ignored. Depending on standardization activities, this may change in the second pilot phase.

Another aspect that is related to save area is the aspect ratio of the FOV that is unknown to the editor. Thus, the editor might want to suggest a target aspect ratio. This will not be done in the first pilot phase but might be introduced later in the project.

2.3. Considerations for an end to end AD and subtitle workflow

This chapter describes the overall approach for ImAc's Audio Description (AD) service from a technical perspective. Based on the initial user requirements that were collected from the focus group tests, this strategy was drafted to provide a technical basis for the AD service.

One result of the ImAc focus group tests shall be mentioned here explicitly: It seems that a major contribution to the user experience may come from an appropriate main audio mix rather than the AD service itself.

Thus, this chapter not only describes the AD service, but includes aspects for audio in general and gives an overview of the influence of various audio formats and playback systems on the immersive (audio) experience.

2.3.1 Audio Systems / Audio Distribution Formats

All audio content (main audio and AD) will be provided in different audio formats in order to support a wide range of user devices. The following audio systems will be supported:

- Stereo: The stereo system will be supported
- Ambisonic: This is a client-side rendering system, which generates binaural audio for headphones. First Order Ambisonic is easy to achieve but does not provide a fine-grained immersion. Higher Order Ambisonic results in a good spatial resolution but has higher costs in hardware requirements on client side.
- Binaural: The 2-channel format for headphones, which allows best spatial resolution. Since this format is already rendered on production side, the audio is static and does not support dynamic changes by head/display rotations. To achieve this, a client-side player logic (targeted for the second pilot phase) is required, which selects the proper streams in real-time. Binaural audio can be played on stereo loudspeakers as well. In comparison to the stereo signal, the user may realize a different "sound colour".

It was considered to deliver object-based audio up to the end user device. This allows for high flexibility regarding the audio scene as well as the playback system, because the audio is rendered on the user device. However, relatively high processing power is needed to perform the client-side rendering. Additionally, the format for delivery of object-based audio has not been established yet. Thus, it was decided to rule out this option for the first pilot phase.

It is planned to use the common audio codec AAC for delivery of all audio streams.

2.3.2 Audio Production Formats

On the production side, object based audio is used preferably: It allows the generation of any output format on demand. Since all audio information is available as “raw data”, the requirements for rendering are relatively high as well as the data traffic. The object based audio scene needs to be rendered into target output formats, which are defined by the producer or within the play-out process.

AD can be edited within the object based audio editor (obA editor) as any other audio object, or – as it is planned in ImAc – externally within an editor specifically addressed for AD productions. In the latter case, the AD track must be imported into an existing audio scene. The obA editor provides an interface for importing AD tracks including their metadata (e.g. regarding positioning and gain).

Stereo and 5.1 productions can be represented by object-based audio as well. For instance, a stereo production would be imported into an audio scene by adding two audio objects – one for each channel. These objects are placed at the location where the loudspeakers would stand in a typical stereo setup. The same approach can be used for 5.1 or any other channel based production.

That way, all typical production formats can be supported in ImAc. This is relevant, since object based audio productions are still rare.

2.3.3 Management of AD file workflow

The management of AD files (and audio files in general) within the premises of a service provider is an important task in order to provide a reliable AD service (to both professional and end users). But the workflows of different providers vary and depend very much on the overall system environment of a company. Thus, it is not in the scope of ImAc to draft a generic architecture for data handling.

However, ImAc installs an exemplary workflow for accessibility data. The Content Manager for accessibility data demonstrates how management of accessibility data can be smoothly included in production and distribution workflows. More details can be found in chapter 3.

The major task here is to add (or mix) AD to the main audio. That may be done automatically outside the editor software. In ImAc, this step is realized by the “AD import interface” of the obA editor that supports all functionalities to embed AD into the audio scene.

After all output formats has been rendered, the AD service needs to be signalized in MPEG DASH such that the player can identify the AD services in the stream. In ImAc the signalization will be triggered by a file naming convention.

2.3.4 Immersive sound and dependencies from playback systems

This section shortly describes the influence of different playback systems on the immersive experience. Only a rough overview can be given here, but it is important to understand that some of the collected user requirements lead to preconditions regarding user hardware and audio formats.

The way that the audio scene should act also depends on the viewing environment that is used. Basically, there are two different kinds of display devices, that will be called “fixed” and “non-fixed”, in the following:

- Fixed display device means that the display device is not rotating or moving. In order to change the viewing direction, another part of the video will be moved into the field of view (FOV). One could say that the video rotates around the user. The user doesn't rotate but always looks in the same direction. Navigation is often done by mouse, arrow keys, swiping on touch display.
- Non-fixed display device means that the user needs to look around (move his head) in order to change the FOV. One could say, that the video does not rotate around the user, but the user will turn to watch different areas of the 360° scene.

Note: Tablets can be used as both fixed and non-fixed displays, since the players often allow two options to change the FOV:

- *Swiping over display (tablet acts like a fixed display)*
- *Using the internal sensors when user turns around with the tablet in hand (tablet acts like a non-fixed display)*

It is assumed that the represented audio scene should align with the video for the best immersive experience. As a result, the kind of display used defines if the audio scene must rotate around the user or not.

Table 3 provides an overview of functionalities and behaviours of different viewing and listening setups.

Table 3 - The functionalities and behaviours of different AD setups

#	Display	Speaker	Notes
1	Fixed display (e.g. PC+browser)	Loudspeaker setup (Stereo, 5.1, …)	Video scene: rotating Playing Ambisonic: The audio scene will rotate together with the video. Playing Stereo or 5.1: The audio scene will not rotate together with the video.

2	Fixed display (e.g. PC+browser)	Headphones	Video scene: rotating Playing Ambisonic: The audio scene will rotate together with the video when the user doesn't turn his head. Additional head movements will not be considered by default. Playing Stereo: The audio scene will not rotate together with the video. Head movements will not be considered by default. Playing binaural sound (no head tracking): Same as "Stereo". Playing binaural sound (with head tracking and corresponding rendering): When head movements and FOV changes can be processed, the audio scene can be reproduced correctly, that means both FOV changes and head movements are considered. The audio scene rotates together with the video even when the user turns his/her head.
3	Non-fixed display (e.g. HMD or Tablet+Sensor)	Loudspeaker setup (Stereo, 5.1, …)	Video scene: not rotating This is a rather unusual scenario, except for one use case: A tablet is used, its internal sensor changes the FOV and audio is played via internal (mono or stereo) speakers. But in this case, the tablet speaker moves as well, which is different from a usual speaker setup. An immersive audio experience is not achieved by this setup and this use case will not be explored in ImAc.

4	Non-fixed display (e.g. HMD or Tablet+Sensor)	Headphones	Video scene: not rotating This is the main viewing environment for ImAc and will lead to the best immersive experience. Head tracking data will be available since it is used to change the FOV. Playing Ambisonic: The audio scene will rotate together with the video. Additional head movements will be considered as well. Playing Stereo: The audio scene will rotate with head movements and will then diverge from the video scene. Playing binaural sound (no head tracking): Same as “Stereo”. Playing binaural sound (with head tracking and corresponding rendering): Same as “Ambisonic” but better localization of audio sources can be achieved.

A binaural stream only represents one viewing direction. To get the best immersive impression, the binaural audio needs to be adapted when viewing direction changes. As a precondition, the viewing angle must be known to the audio processor. Basically, there are two approaches to reproduce binaural audio for the current viewing direction:

- A stream for each viewing direction is pre-rendered and provided (typical resolution: 1°)
- The binaural signal is rendered dynamically

Note: The delay between head movement and corresponding change of the binaural signal must be small (around 40ms). Otherwise the impression of a stable audio scene will break.

The different ImAc formats that are supported were chosen to provide acceptable audio for all viewing environments. In general, playing binaural audio on headphones, where the binaural sound changes according to the user’s head movements, will deliver the best user experience. The support of this feature is targeted for the second pilot phase.

2.3.5 User Preferences

The following user preferences regarding AD presentation will be supported in ImAc:

- The user can choose from different gain levels for the AD channel (in relation to main mix).
- The user can choose from different locations for the AD speaker (i.e. from which direction the sound will come from).

There are two options regarding the realization of this feature that were discussed in ImAc:

1. Deliver the AD channel as a separate stream and mix it with the main audio on the user device.
2. Pre-render all variations a user can choose from and provide several streams.

Client-side mixing allows for more flexibility and is definitely the way to go when object based audio is brought up to the client device. Since in ImAc object based audio is only used at server side (at least for the first pilot phase), the different user preferences will be pre-mixed by the service provider and exposed to the client within the distribution (e.g. MPEG-DASH manifest). The signalization of the different options that are available is done by naming convention of the streams or custom DASH extensions.

2.3.6 Using Multiple AD tracks

Providing multiple AD tracks allows for more than one description of the current video content to be active at the same time. If an AD track is played or muted, depends on the current viewing direction of the user.

This feature will not be defined at this stage of the project, because a more detailed use case description is not available yet. The feature may be added later in the project for the second pilot phase.

2.4 Considerations for an end to end Signer workflow

The signer service will be processed as a side video track. The technical details of signer service will be defined in a later stage of the project and described in the second iteration of this document.

3. ARCHITECTURAL REPRESENTATION

This document presents the architecture as a series of views, based on the '4+1' View Model of software architecture [1].

A set of scenarios provide a description of the use-case view of the software architecture describing different situations and/or use cases that represent some significant, central functionality.

The logical view describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers.

The process view captures the concurrency and synchronization aspects of the design, and describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. It also describes the allocation of objects and classes to tasks.

The deployment view of the architecture describes the various physical nodes for the most typical platform configurations. It also describes the allocation of tasks (from the Process View) to the physical nodes.

Based on the '4+1' model we do not provide a Development View as the development team structure and software management is discussed in the project proposal.

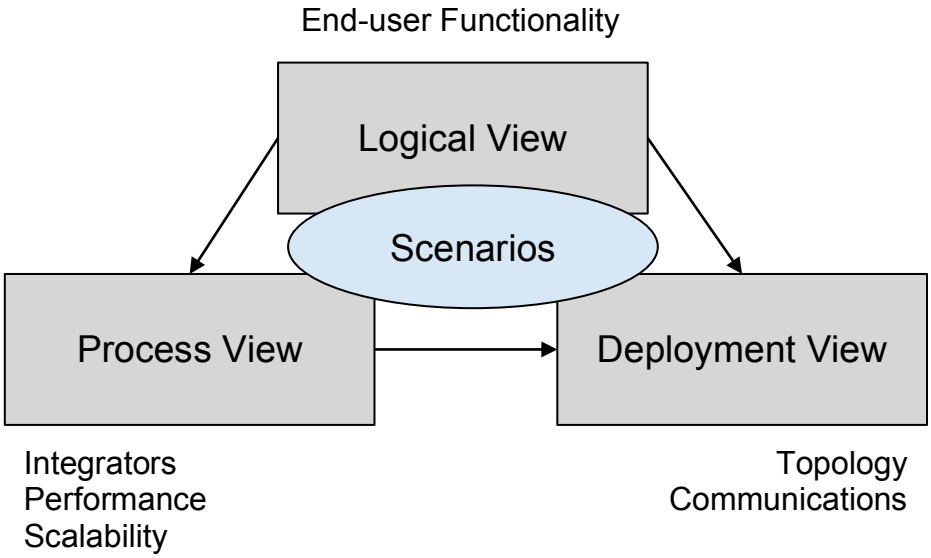

Figure 2 - The structure of the ImAc architectural representation

3.1. Scenarios

Different scenarios are described using a use-case view of the software architecture in order to highlight significant and central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture. Table 4 shows an overview of the scenarios defined within the project.

Table 4 - Overview of Scenarios

Production Editors	S1	Production of New Subtitles
	S2	Subtitle Verification and correction
	S3	Production of Audio Description
	S4	Audio Description Verification and Correction
	S5	Production of Sign Language
	S6	Sign Language Verification and Correction
	S7	Object Based Audio Editing
Accessibility Content Manager	S8	Assignment of videos for the accessibility content production
	S9	Getting an Accessibility Content file
	S10	Generating Different audio formats with an audio renderer
Content Packaging and Distribution	S11	Preparation of Contents
	S12	Distribution
Player	S13	Consumption of media contents

3.1.1 Production Editors

In all the following use cases the user is a professional user.

3.1.1.1 New Subtitling

Table 5 – Scenario for production of new subtitles

S1.1	User accesses the production web page via the web browser.
S1.2	User enters username and password.
S1.3	A window with the list of videos to be subtitled by the user appears.
S1.4	User selects the video to be subtitled and presses the edit button.
S1.5	The web-subtitling editor opens with the video in the preview window and with no subtitles.
S1.6	User creates a new subtitle with all the associated metadata (timecode, character, angle of view, etc.). For that user can use the video player buttons and mouse to move around the video in order to find the entry and exit points of the subtitle, find the character (by video panning) and listen to the corresponding character.
S1.7	The subtitle is simulated over the preview video window.
S1.8	User moves to the next empty subtitle by pressing the corresponding key.
S1.9	User repeats the same process with the rest of the subtitles.
S1.10	User checks the subtitling restrictions by pressing the corresponding button.
S1.11	If a subtitle doesn't comply with the restrictions, a message appears for the user to correct it before continuing with the rest of the checking process.
S1.12	User corrects the subtitle to comply with the restrictions.
S1.13	User repeats the same process until all subtitle items are compliant. In this last case, the simulation button is pressed.
S1.14	User presses the simulation button to verify the final result.
S1.15	User uses the preview video player buttons to move around and see the result accordingly. For instance if the user presses the play button the subtitles will be displayed and deleted

	over the preview video in their respective times along with the video playback so as to simulate the final result.
S1.16	User can also press the HMD simulation by pressing the corresponding button.
S1.17	User puts on the HMD to watch the final result.
S1.18	User can move up and down the subtitles if specific ones need to be corrected.
S1.19	The subtitling is saved automatically in the Accessibility Content Manager

3.1.1.2 Subtitle verification and correction

Table 6 - Scenario for subtitle verification and correction

S2.1	User accesses the production web page via the web browser.
S2.2	User enters username and password.
S2.3	A window with the list of videos to be verified by the user appears.
S2.4	User selects the video to be verified and presses the edit button.
S2.5	The web-subtitling editor opens with the video in the preview window and with the subtitling.
S2.6	User checks the subtitling restrictions by pressing the corresponding button.
S2.7	If a subtitle doesn't comply with the subtitling restrictions a message appears for the user to correct it before continuing with the rest of the checking process.
S2.8	User corrects the subtitle to comply with the subtitling restrictions.
S2.9	User repeats the same process until there's no subtitle that does not comply. In this last case the simulation button activates.
S2.10	User presses the simulation button to verify the final result.
S2.11	User uses the preview video player buttons to move around and see the result accordingly. For instance if the user presses the play button the subtitles will be shown and deleted over the preview video in their respective times along with the video playback so as to simulate the final result.
S2.12	If the subtitling and the video are not in sync, user presses the offset button and enters the right entry time (TCin) that is required for the current subtitle. The same offset is applied to the rest of subtitles times.
S2.13	User presses the simulation button again to verify the final result.
S2.14	User can also press the HMD simulation by pressing the corresponding button.
S2.15	User puts on the HMD to watch the final result.
S2.16	User can move up and down the subtitles if specific ones need to be corrected.
S2.17	The subtitling is updated automatically in the Accessibility Content Manager database.

3.1.1.3 New Audio Description

Table 7 - Scenario for production of Audio Description

S3.1	User accesses the production web page via the web browser.
S3.2	User enters username and password.
S3.3	A window with the list of videos to be audio described by the user appears.
S3.4	User selects the video to be audio described and presses the edit button.

S3.5	The web audio description editor opens with the video in the preview window and with no audio description.
S3.6	User creates a new audio description segment with all the associated metadata (TCs, character, angle of view, etc.). The user can use the video player buttons and mouse to move around the video in order to find the entry and exit points of the audio description, find the character (by video panning) and watching the corresponding scene.
S3.7	User moves to the next empty audio description segment by pressing the corresponding key.
S3.8	User repeats the same process with the rest of the audio description segments.
S3.9	User checks the audio description restrictions by pressing the corresponding button.
S3.10	If an audio description segment doesn't comply with the restrictions a message appears for the user to correct it before continuing with the rest of the checking process.
S3.11	User corrects the audio description segment to comply with the restrictions.
S3.12	User repeats the same process until there's no audio description segment that does not comply. In this last case the simulation button activates.
S3.13	User presses the simulation button to verify the final result.
S3.14	User uses the preview video player buttons to move around and see the result accordingly. For instance if the user presses the play button the audio description segments will be playback mixed with the preview video audio in their respective times along with the video playback so as to simulate the final result.
S3.15	User can also press the HMD simulation by pressing the corresponding button.
S3.16	User puts on the HMD device to watch the final result.
S3.17	User can move up and down the audio description segment if specific ones need to be corrected.
S3.18	The audio description is saved automatically in the Accessibility Content Manager database.
S3.19	User might export a json file for description of AD position/gain for the obA Editor together with raw AD audio files.

3.1.1.4 Audio Description Verification and Correction

Table 8 – Scenario for Audio Description Verification and Correction

S4.1	User accesses the production web page via the web browser.
S4.2	User enters username and password.
S4.3	A window with the list of videos to be verified by the user appears.
S4.4	User selects the video to be verified and presses the edit button.
S4.5	The web editor opens with the video in the preview window and with the audio description.
S4.6	User checks the audio descriptions restrictions by pressing the corresponding button.
S4.7	If an audio description doesn't comply with the audio description restrictions a message appears for the user to correct it before continuing with the rest of the checking process.
S4.8	User corrects the audio description to comply with the audio description restrictions.
S4.9	User repeats the same process until there's no audio description that does not comply. In this last case the simulation button activates.
S4.10	User presses the simulation button to verify the final result.
S4.11	User uses the preview video player buttons to move around and see the result accordingly.

	For instance, if the user presses the play button the audio description will be played and deleted over the preview video in their respective times along with the video playback so as to simulate the final result.
S4.12	If the audio description and the video are not in sync, user presses the offset button and enters the right entry time (TCin) that is required for the current audio description. The same offset is applied to the rest of audio description times.
S4.13	User presses the simulation button again to verify the final result.
S4.14	User can also press the HMD simulation by pressing the corresponding button.
S4.15	User puts on the HMD device to watch the final result.
S4.16	User can move up and down the subtitles if specific ones need to be corrected.
S4.17	The audio description is updated automatically in the Accessibility Content Manager database.

3.1.1.5 New Sign Language

Table 9 - Scenario for Production of new sign language

S5.1	User accesses the production web page via the web browser.
S5.2	User enters username and password.
S5.3	A window with the list of videos to be signed by the user appears.
S5.4	User selects the video to be signed and presses the edit button.
S5.5	The web sign language editor opens with the video in the preview window and with no sign language.
S5.6	User creates a new interpreter segment with all the associated metadata (TCs, character, angle of view, etc.). For that user can use the video player buttons and mouse to move around the video in order to find the entry and exit points of the subtitle, find the character (by video panning) and listen to the corresponding character.
S5.7	The interpreter video is shown in an independent window next to the preview video player window.
S5.8	User moves to the next empty interpreter segment by pressing the corresponding key
S5.9	User repeats the same process with the rest of the interpreter segments.
S5.10	User checks the sign language restrictions by pressing the corresponding button.
S5.11	If an interpreter segment doesn't comply with the restrictions, a message appears for the user to correct it before continuing with the rest of the checking process.
S5.12	User corrects the interpreter segment to comply with the restrictions.
S5.13	User repeats the same process until there's no interpreter segment that does not comply. In this last case the simulation button activates.
S5.14	User presses the simulation button to verify the final result.
S5.15	User uses the preview video player buttons to move around and see the result accordingly. For instance if the user presses the play button the interpreter video will be played back in their respective times in its window next to the video playback so as to simulate the final result.
S5.16	User can also press the HMD simulation by pressing the corresponding button.
S5.17	User puts on the HMD device to watch the final result.
S5.18	User can move up and down the interpreter segment if specific ones need to be corrected.
S5.19	The sign language is saved automatically in the Accessibility Content Manager database.

3.1.1.6 Sign Language Verification and Correction

Table 10 - Scenario for sign language verification and correction

S6.1	User accesses the production web page via the web browser.
S6.2	User enters username and password.
S6.3	A window with the list of videos to be verified by the user appears.
S6.4	User selects the video to be verified and presses the edit button.
S6.5	The web editor opens with the video in the preview window and with the subtitling.
S6.6	User checks the sign language restrictions by pressing the corresponding button.
S6.7	If a sign language doesn't comply with the sign language restrictions a message appears for the user to correct it before continuing with the rest of the checking process.
S6.8	User corrects the sign language to comply with the sign language restrictions.
S6.9	User repeats the same process until there's no sign language that does not comply. In this last case the simulation button activates.
S6.10	User presses the simulation button to verify the final result.
S6.11	User uses the preview video player buttons to move around and see the result accordingly. For instance if the user presses the play button the sign language will be shown and deleted over the preview video in their respective times along with the video playback so as to simulate the final result.
S6.12	If the sign language and the video are not in sync, user presses the offset button and enters the right entry time (TCin) that is required for the current subtitle. The same offset is applied to the rest of sign language times.
S6.13	User presses the simulation button again to verify the final result.
S6.14	User can also press the HMD simulation by pressing the corresponding button.
S6.15	User puts on the HMD to watch the final result.
S6.16	User can move up and down the sign language segment if specific ones need to be corrected.
S6.17	The sign language is updated automatically in the Accessibility Content Manager database.

3.1.1.7 Object-Based Audio Editor

Table 11 - Scenario for object based audio editing

S7.1	User opens the object-based audio editor.
S7.2	User opens object-based audio scene or creates a new scene.
S7.3	User adds object-based audio objects to the scene (if necessary).
S7.4	User edits audio objects (if necessary).
S7.5	User manually edits or add tracks.
S7.5.1	*User downloads AD track(s) from the Content Manager.*
S7.5.2	*User imports AD track (Result from AD editor, with spatial metadata included).*
S7.5.3	*User exports object-based audio scene.*
S7.6	The spatial metadata is transmitted via a json file.
S7.6.1	*User exports json configuration of AD objects from AD Editor.*
S7.6.2	*User exports raw audio files of AD tracks from AD Editor.*

S7.6.3	*User exports obA Scene file (ADM file) from obA Editor without the AD.*
S7.6.4	*User merges exports from AD Editor and obA Scene into one scene with standalone software.*

3.1.2 Accessibility Content Manager

3.1.2.1 Assignment of videos for the accessibility content production

Table 12 - Scenario for Assignment of videos for accessibility content production

S8.1	User accesses the accessibility content manager web page via the web browser.
S8.2	User enters username and password.
S8.3	A window with the list of assets with its corresponding accessibility files appears.
S8.4	User presses the new asset to upload the video to be subtitled, audio described or signed.
S8.5	User selects the new asset, edits metadata and presses the assign production button (subtitling, audio description or sign language).
S8.6	User assigns the corresponding accessibility content production to an operator (professional user that will have to produce the accessibility content).
S8.7	User repeats the process for the rest of videos.

3.1.2.2 Getting an Accessibility Content file

Table 13 – Scenario for getting accessibility content files

S9.1	User accesses the accessibility content manager web page via the web browser.
S9.2	User enters username and password.
S9.3	A window with the list of assets with its corresponding accessibility files appears.
S9.4	User presses the find button and enters the criteria for the search.
S9.5	A list of assets that comply with the criteria is displayed.
S9.6	User selects the asset and executes the download of the corresponding accessibility content file (subtitling, audio description or sign language).

3.1.3 Content Packager and Distribution

3.1.3.1 Generating Different audio formats with an audio renderer

Table 14 - Scenario for generating different audio formats

S10.1	User loads object-based audio scene.
S10.2	User loads settings for rendering process and export.

S10.3	User edits settings for rendering process and export (if necessary).
S10.4	User defines export formats (first order ambisonics, binaural, stereo,...).
S10.5	User starts rendering process.

3.1.3.2 Preparation of contents for the end user visualization

Table 15 – Scenario for Preparation of Contents for Distribution

S11.1	Subtitling, audio description and sign language within the corresponding audio-visual contents are packaged
S11.2	Distribution of the packaged contents to the client players.

3.1.3.3 Distribution

Table 16 – Scenario for Distribution

S12.1	**VoD**: the content will be prepared as files on Content Delivery Network (CDN) storage ready to be distributed on a video on demand request.
S12.2	**VoD with TV linear distribution**: the content will be distributed to be consumed through companion screen synchronously to a linear TV content. In this case, a server may be required to package live or, as a more likely scenario, the content can be pre-packaged to be broadcast at a given time. Then the content is pushed to the CDN (may require authentication tokens which depend on the CDN of the broadcaster).

3.1.4 Player

Table 17 - Scenario for playback

S13.1	Scenario 1: User accesses to a website in which a list of accessible 360º video contents (i.e., immersive contents enriched with accessibility assets) is available.
S13.2	Scenario 2: User is watching traditional TV, while it receives a notification about the availability of related immersive and accessible 360º video contents.
S13.3	User could use HMDs, smartphones and tablets for the playback of the immersive + accessible contents in both scenarios. In scenario 1, the use of PCs (laptops, desktops…) is also possible
S13.4	User presses the proper button/link to start the playback of the immersive + accessible contents. The playback can also automatically start in scenario 2.
S13.5	User can enable the accessibility services, which allows an adaptive and personalized presentation of the accessibility assets (subtitles, audio subtitles, sign language, and audio description).
S13.6	User can dynamically activate/deactivate the presentation of the available accessibility assets, as well as to set the available personalisation options for each of them.
S13.7	During playback, user can freely explore the 360º area (e.g., by moving the head, using the touch screen, mouse…), and the presentation of the accessibility assets will be adapted accordingly.
S13.8	In case of multi-screen scenarios, a time-aligned presentation of the media playback across to involved devices will be provided, so coherent sessions can be experienced.
S13.9	When the video playback is finished, user can decide to watch it again, to watch other one, or leave the ImAc accessibility service.

3.2. Logical Views

This section provides a description of the logical view of the architecture. We describe the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. It also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

The logical view of the ImAc system is comprised of the main packages: Editor Tools (WP4) Content Manager (T3.2), Content Packager / Distributor (T3.3) and Player (T3.5)

3.2.1 Production Editors

Figure 3 provides a logical view diagram for the subtitling editor, Figure 4 provides a logical view diagram for the audio description editor and Figure 5 provides a logical view diagram for the sign language editor.

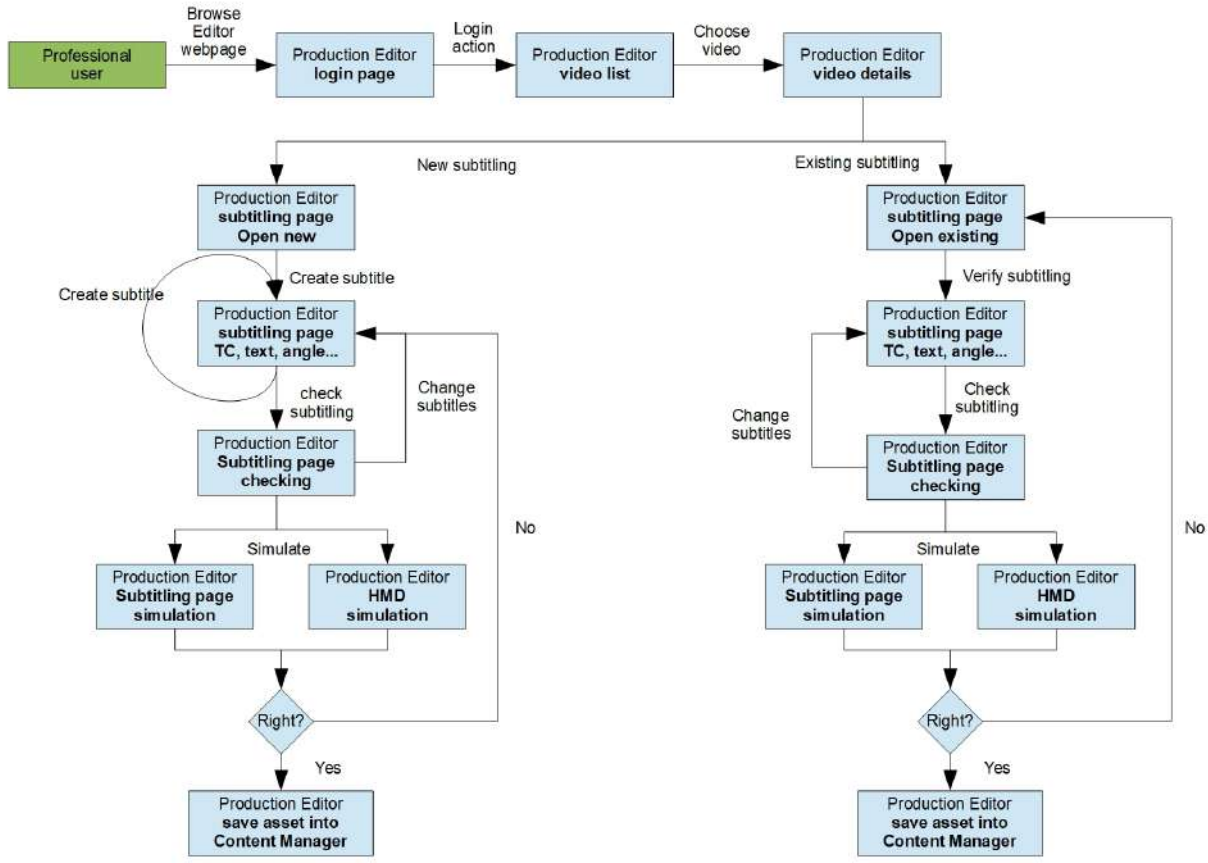

Figure 3 - Logical View Diagram - Subtitling Editor

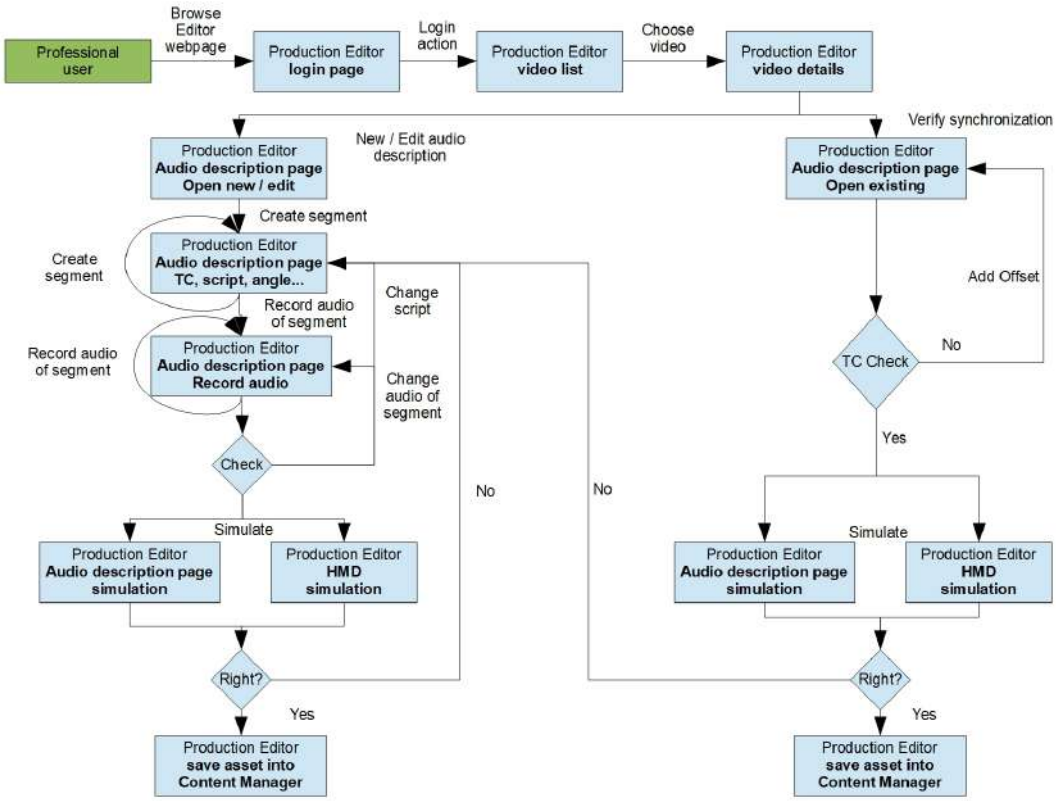

Figure 4 - Logical View Diagram - Audio Description Editor

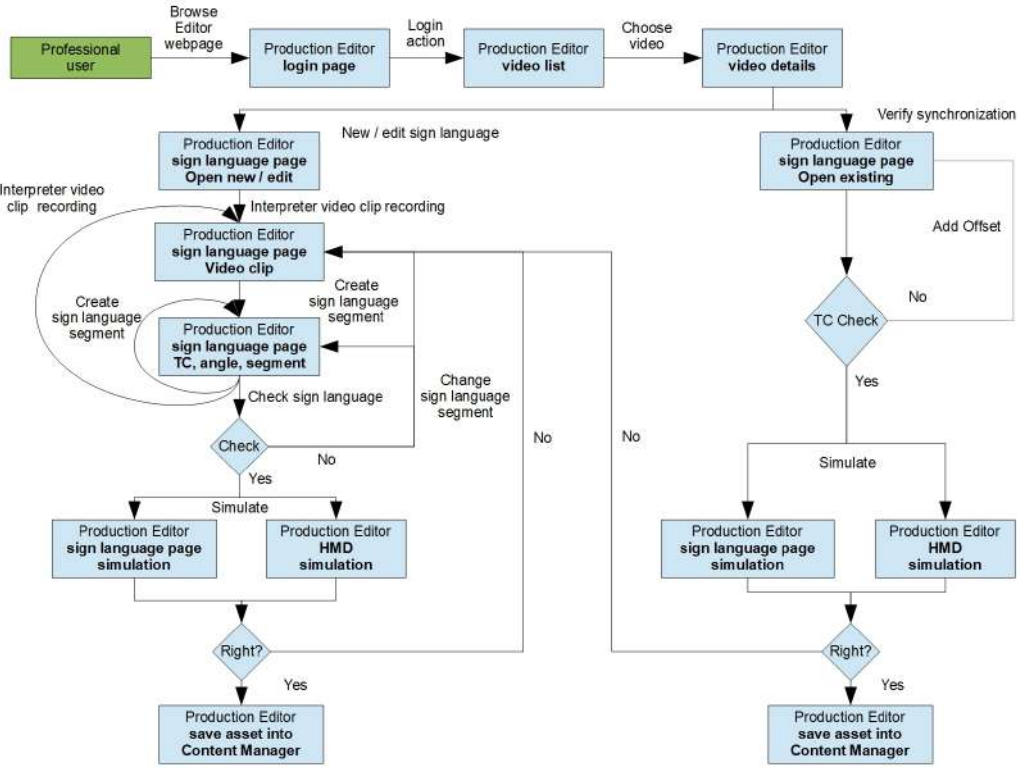

Figure 5 - Logical View Diagram Sign Language Editor

3.2.2 Accessibility Content Manager

Figure 6 provides a logical view diagram for the accessibility content manager.

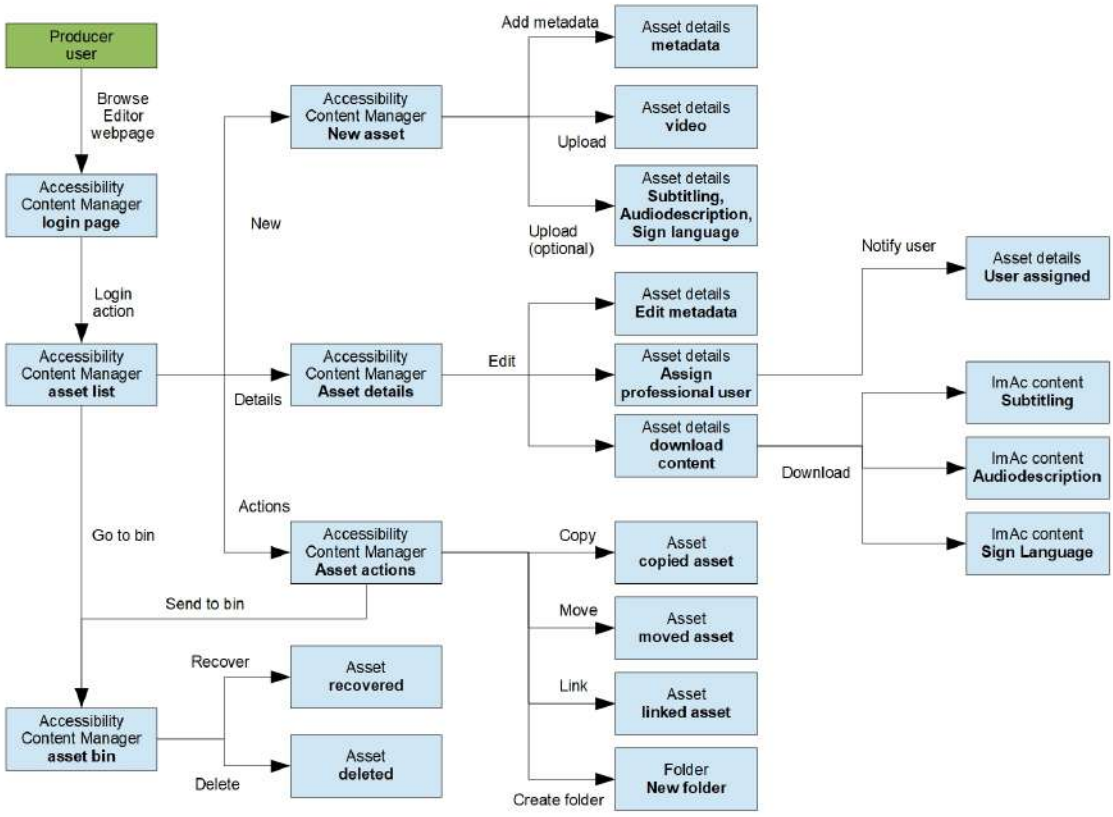

Figure 6 - Logical View Diagram - Accessibility Content Manager

3.2.3 Content Packager and Distribution

Figure 7 shows a logical View of the content packager and distribution module. This module will communicate with the Content Manager in a close way since the Content Manager will pilot it. The ImAc project will define protocols to ensure communications between the modules following the best practices of the industry.

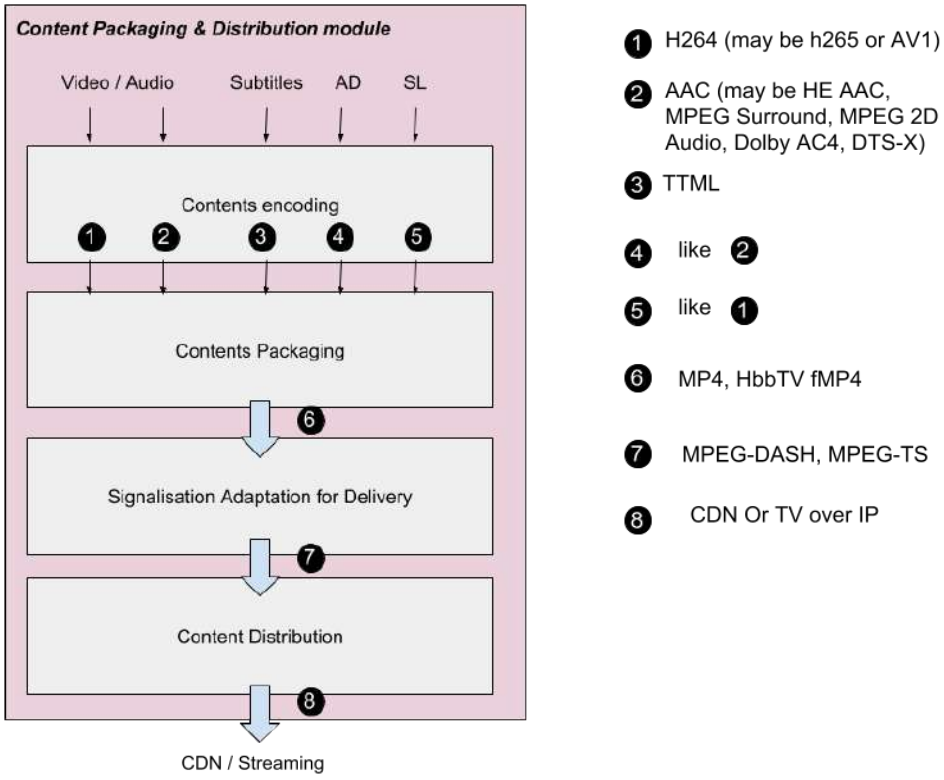

Figure 7 - Logical View Diagram - Content packager and distribution module

This module will match four main required functionalities:

- Firstly, it will encode the content (compress the audio and video).
  - Video: h.264. Maybe h.265 or AV1 depending on technical and standardization advances.
  - Audio (including AD): AAC. Maybe other codecs or extensions (HE-AAC/MPEG Surround, MPEG 2D Audio, Dolby AC4, DTS-X) depending on SDK availability since most of these codecs are proprietary or heavily patented.
  - Subtitles: TTML (v1, IMSC1 profiles) (or WebVTT if standardization occurs).
  - SL: (MP4)
- Secondly, it will package the content inside a suitable container for delivery, and it will ensure that the content is segmented appropriately.
  - TV over IP: MP4
  - TV: HbbTV..

    Note: we may want to use the CMAF ISOBMFF application format.
- Thirdly, this module will ensure a proper delivery that includes the signalling of the metadata according the editors' wishes.
  - MPEG-DASH (DASH-IF and HbbTV industry profiles)
  - MPEG-TS (according to HbbTV)
- Finally, it will push the content to the appropriate network (whether using streaming protocols, or file-based caching infrastructures).
  - CDN only.
  - TV over IP for TV.

3.2.4 Player

Figure 8 shows a high-level overview of the logical view for the player. It illustrates the interactions between the user and the player, as well as of how the presentation of the immersive and accessibility contents can be dynamically activated/deactivated and adapted according to tracking functions (e.g. users' movements, interactions…) and to the setting of the available personalisation features.

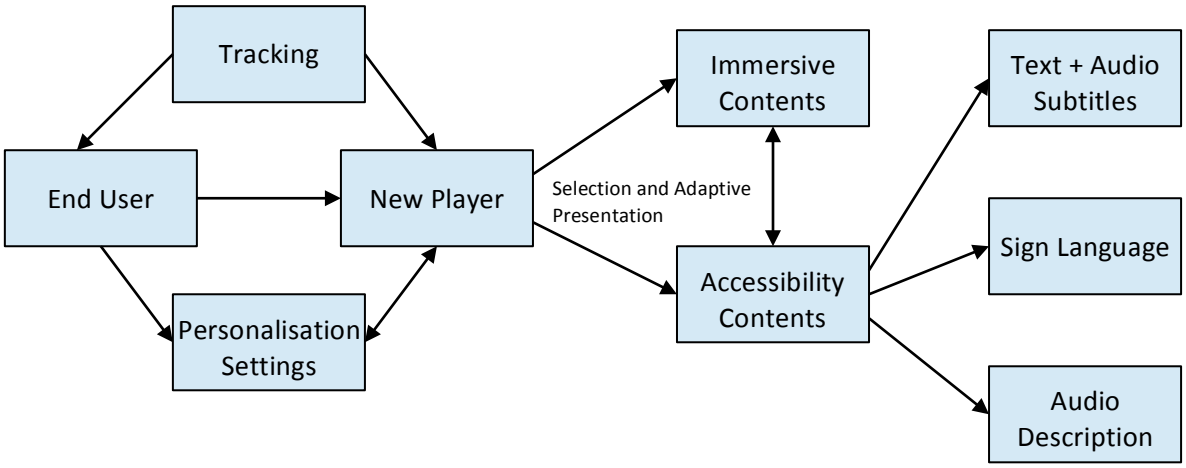

Figure 8 - Logical View for the Player

In relation to this, Figure 9 illustrates the main sub-systems, layers, modules and libraries that compose the ImAc player, together with the relationships and interactions between these components. The names of the involved JS libraries implementing the functionalities of these components are also indicated. By using these libraries, the instances of the classes implementing each of the ImAc player's requirements are created. In the figure, AV stands for Audio+Video, V for Video, A for Audio, SL for Sign Language, S for Subtitles, and DVB-CSS for Digital Video Broadcasting - Companion Screens and Streams.

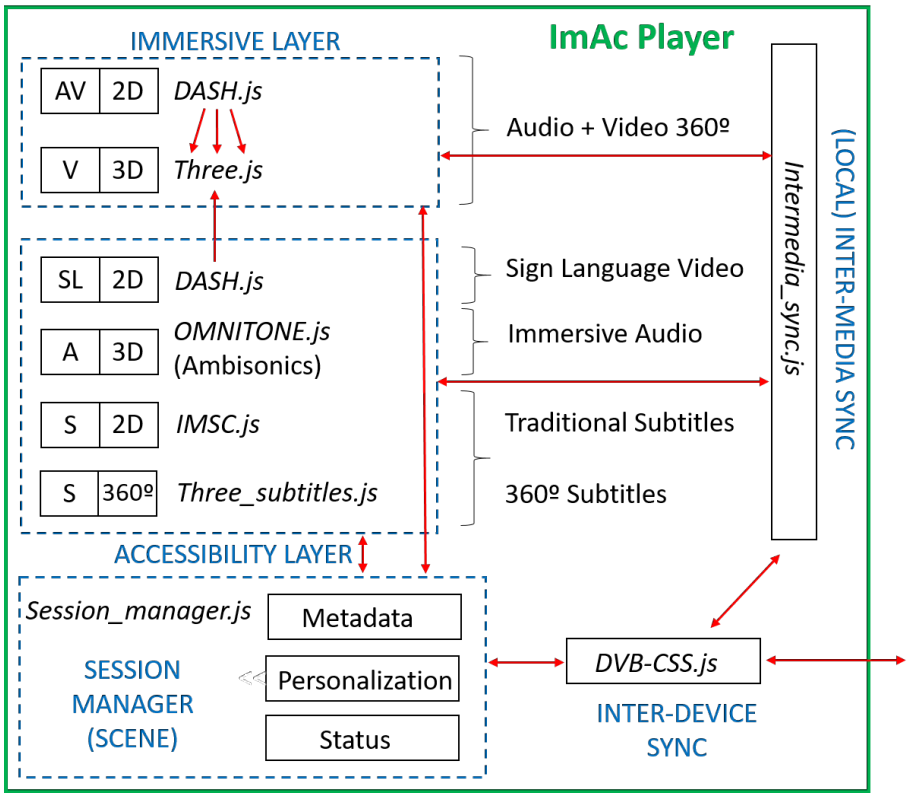

Figure 9 - Sub-systems, layers and modules composing the player

3.3. Process Views

This section provides a description of the process view of the architecture. It describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. It also describes the allocation of objects and classes to tasks.

3.3.1 Production Editors processes

3.3.1.1 Subtitling editor processes

- Video player process: render the received 360 video from the server for the preview player.
- Subtitling simulation process:
  - Each time a subtitle is edited it is previewed on top of the video.
  - When the subtitling is finished a simulation of the complete subtitling is done on top of the video while playing back the video.
- HMD simulation process: when the subtitling is finished a simulation of the complete subtitling is done on the HMD.

3.3.1.2 Audio description editor processes

- Video player process: render the received 360 video from the server for the preview player.
- Audio description simulation process:

    - Each time an audio description segment is edited a mixed audio preview can be played back for the verification.
    - When the audio description is finished a simulation of the complete audio description is done along the playback of the video.
- HMD simulation process: when the audio description is finished a simulation of the complete audio description is done on the HMD.

3.3.1.3 Sign language editor processes

- Video player process: render the received 360 video from the server for the preview player.
- Sign language simulation process:
    - Each time a sign language segment is edited a preview of the corresponding video clip can be played back for the verification.
    - When the sign language is finished a preview of the sign language in a separate window is done along the playback of the main video.
- HMD simulation process: when the sing language is finished a simulation of the complete sign language is done on the HMD.

3.3.1.4 Processes tasks of object-based audio editor

- Object-based audio editor task: Create or edit object based audio scenes. This includes import of various audio formats (e.g. stereo, 5.1, Ambisonics) to convert them into object based audio scene format.
- Content uploader task (push content to audio renderer): Export final audio scene and transfer it to the audio renderer.

3.3.2 Accessibility Content Manager processes

Accessibility Content Manager processes:

- Video uploading process: when a video is uploaded it is stored in the database with the respective metadata.
- Accessibility content uploading process: when accessibility content file is uploaded it is associated with the corresponding video in the database.
- Accessibility content edition process: when a accessibility content file is uploaded from the editors, it is updated in the database.
- Accessibility content delivery: when accessibility content file is requested from the web service, it is converted into the corresponding file format and delivered.

Content Packager and Distribution

3.3.3 Player processes

End-users create an instance of the media player by opening a web browser, typing the target URL and selecting the appropriate contents (in S13.1) or by directly associating a companion device with a main TV (in S13.2). The selection or direct playback of immersive contents will enable the Immersive Layer of the player, which includes the functionalities for creating 360º

scenes, by using *three.js* library, from instances of traditional 2D media players, by using the *DASH.js* library. The Accessibility Layer will be enabled through controls of the User Interface, although it may be also automatically enabled based on the user's profile. That layer includes modules for the presentation of the considered accessibility assets, namely: textual and audio subtitles, sign language videos, and audio description (see Figure 10).

When more than one media component is being simultaneously presented, their spatial and temporal relationships must be preserved during playback. A new developed *intermedia_sync.js* library needs to be developed to achieve this goal.

Moreover, key functionalities of the player will be provided by the Session Manager module, implemented in the session_manager.js library. That module will store relevant information about the session, such as the metadata obtained from the Content Manager (e.g., describing the available contents, their relationships, etc.), the contents being currently presented, the available personalisation options together with the current settings, as well as the status of the session (e.g., elapsed time, duration, other active devices...).

3.4. Deployment Views (Physical Views)

A description of the deployment view of the architecture describes the various physical nodes for the most typical platform configurations. This section also describes the allocation of tasks (from the Process View) to the physical nodes.

This section is organized by physical network configuration; a deployment diagram, followed by a mapping of processes to each processor, illustrates each such configuration.

3.4.1 Production Editors

The production editors require a Desktop PC for object-based audio editor.

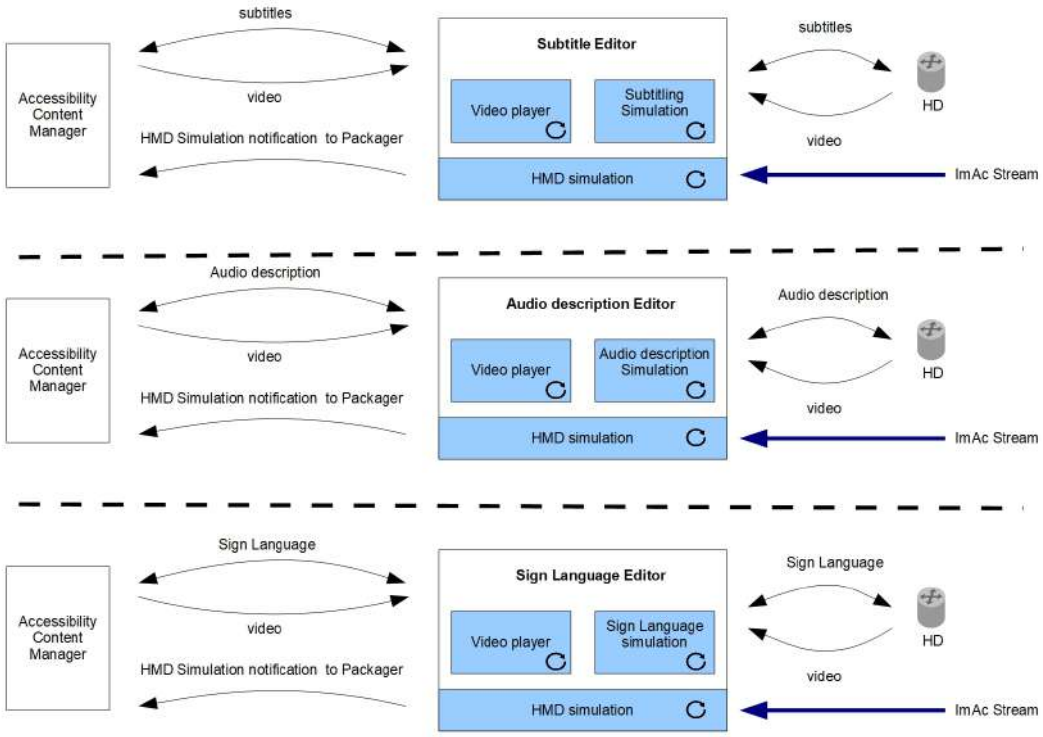

Figure 10 - Deployment View Diagram - Production Editors

The object based audio editor runs as stand alone software on a Desktop PC.

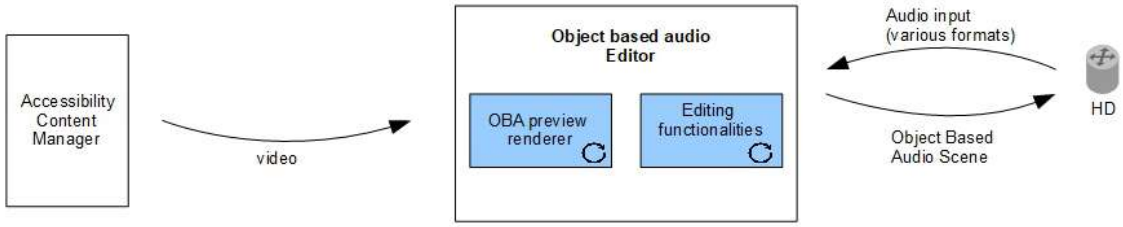

Figure 11 - Deployment View Diagram - Object based audio renderer

3.4.2 Content Manager

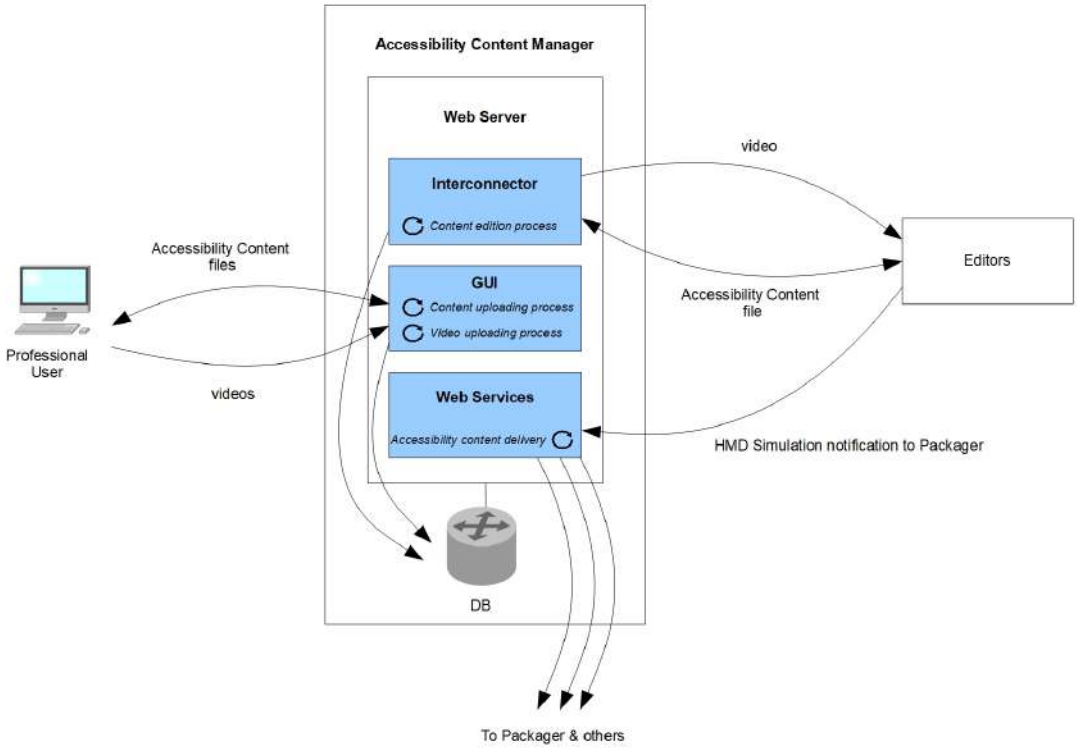

Figure 12 - Deployment View Diagram - Accessibility Content Manager

3.4.3 Content Packager / Distribution

The following servers are required for the deployment of the content packager and distributer as implemented in T3.3:

- FTP for shared folder.
- Access to Amazon Cloud Server for audio rendering
- Video encoder and DASH packager (MSE)
    - Offline for VoD or “fake” linear.
- Question: is it part of 3.4.3 or 3.4.2 Signer preparation
- Server for Web Service
- CDN

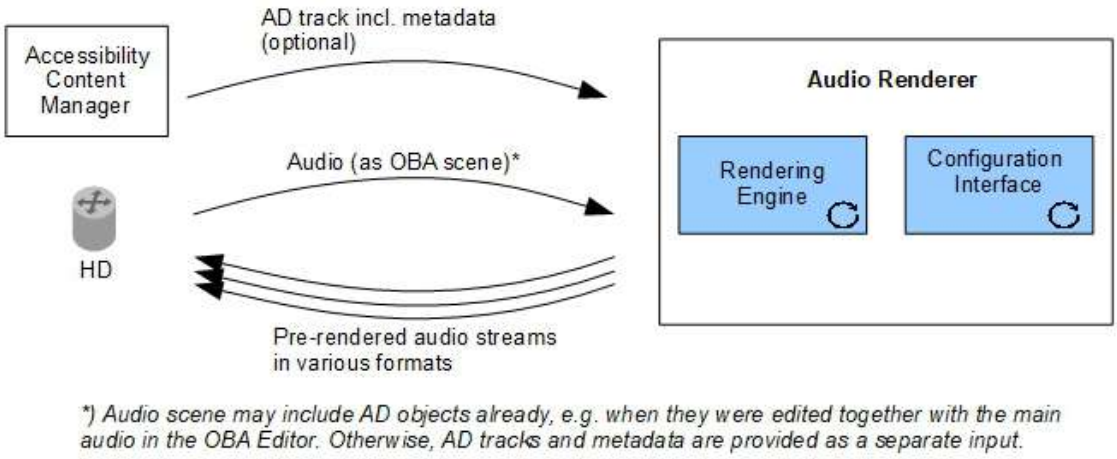

Figure 13 - Deployment View Diagram - Audio Renderer

3.4.4 Player

Figure 14 illustrates the involved physical nodes or entities and their configurations on the consumer side. The considered scenarios can involve either only (maybe independent) companion devices (presenting both immersive and accessibility contents) or a single main TV (presenting traditional or immersive contents) and one or multiple (n) companion devices (presenting both immersive and accessibility contents).

The project contemplates two development phases. In the first one, only web scenarios will be considered. This means that both the main TV and companion devices will include a web-based player for the presentation of the ImAc media contents. In case that main and companion devices form part of a shared session, proper discovery, association and app launching mechanisms will be provided. In addition, the DVB-CSS protocols, adopted by HbbTV standard, will be used to guarantee synchronized playback processes across the involved devices. This is illustrated on the left part of Figure 14. The second development phase will only be considered given the availability of equipment supporting the latest version of HbbTV standard, v 2.0.1. This will allow synchronizing the playback of broadcast contents presented on the main TV with the playback of broadband related contents presented on the companion devices. In such a case, the discovery, association and app launching mechanisms are also natively supported by HbbTV 2.0.1, and thus being provided by the involved equipment / devices.

The communications and interactions between the consumer devices and the Content Manager to be aware of the available contents, select them and retrieve them are shown in Figure 14.

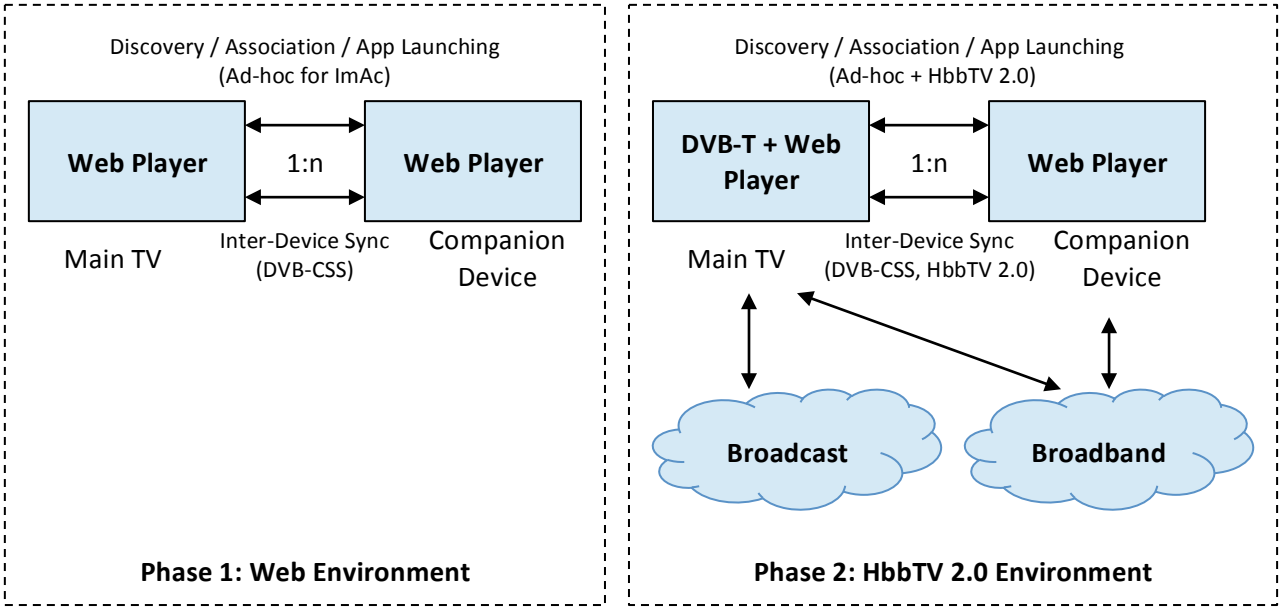

Figure 14 - Physical Nodes and configurations in the consumer side

3.4.5 Complete ImAc System

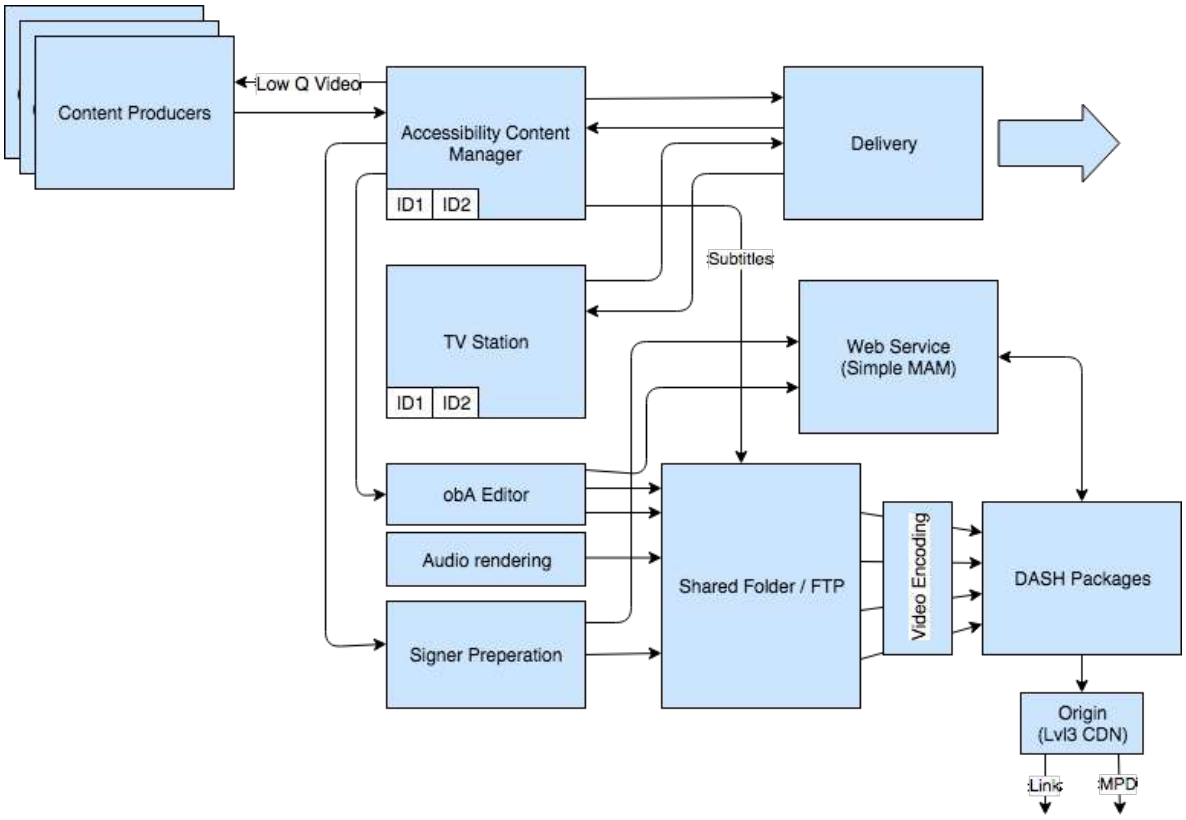

Figure 15 - Deployment View Diagram - complete ImAc system

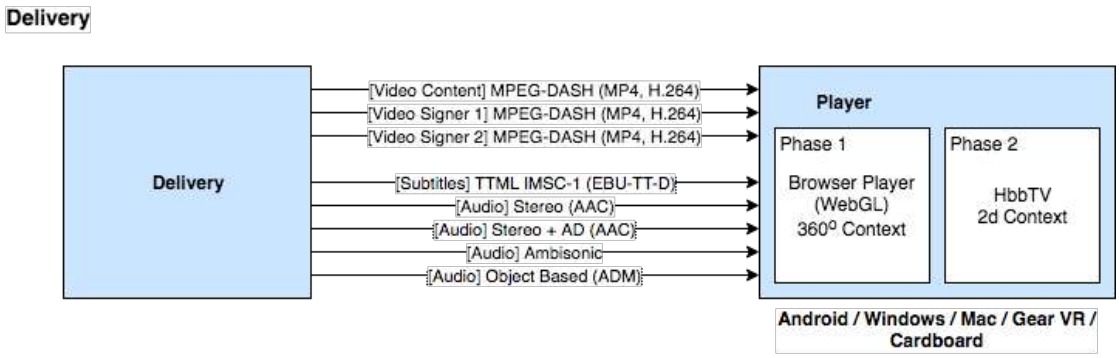

Figure 16 - Deployment View Diagram – Delivery

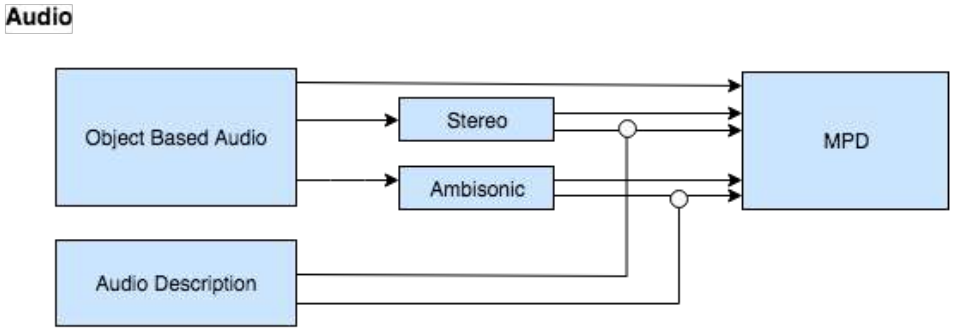

Figure 17- Deployment View Diagram – Audio

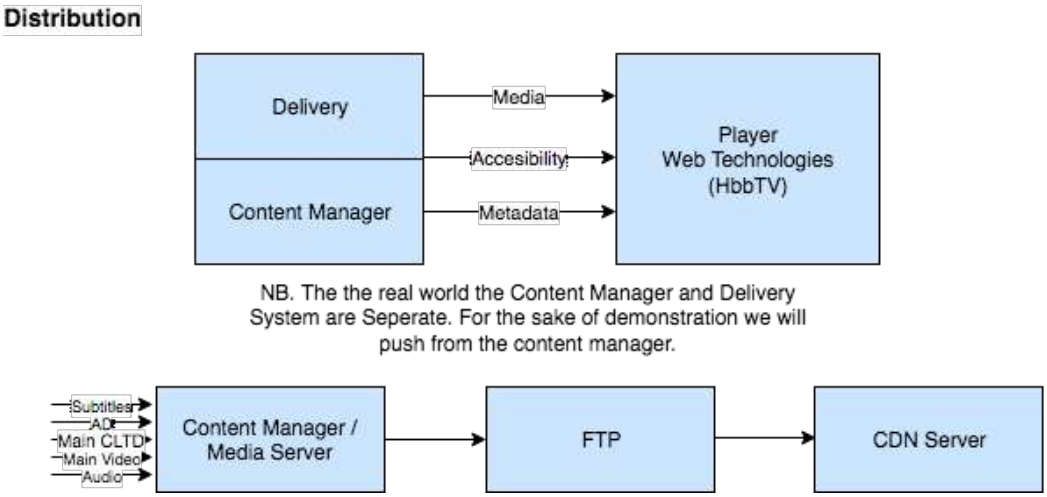

Figure 18 - Deployment View Diagram - Distribution

4. DEPENDENCIES

4.1. Player Dependencies

The player is built on a number of existing tools. These are detailed in table 18.

Table 18 - Player Dependencies

Subtitles		IMSC.js
Video	2D	DASH.js
	360	Three.js (WebGL)
Audio	360	Omnitone.js (Ambisonic)
Sync		DVBCSS.js
Scene		SESSION_MANAGER.js
(Local) Inter Media Sync		intermedia_sync.js
Inter Device Sync		DVB-CSS.js

4.1.1 IMSC.js

https://github.com/sandflow/imscJS

imscJS is a JavaScript library for rendering IMSC Text and Image Profile documents to HTML5. IMSC1 is a profile of TTML designed for subtitle and caption delivery worldwide.

4.1.2 DASH.js

https://github.com/Dash-Industry-Forum/dash.js/wiki

A reference client implementation for the playback of MPEG DASH contents via JavaScript and compliant browsers.

4.1.3 Three.js

Three.js is a cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser, without the need of proprietary plugins. Three.js uses WebGL. It is used to compose the 360º environments from traditional 2D videos processed by *DASH.js*.

4.1.4 Omnitone.js

https://github.com/GoogleChrome/omnitone

Omnitone is a robust implementation of ambisonic decoding and binaural rendering written in Web Audio API. Its rendering process is powered by the fast native features from Web Audio API (GainNode and Convolver), ensuring the optimum performance.

4.1.5 Intermedia_sync.jss

Library that will handle the temporal synchronization between the playback of all considered immersive and accessible contents within each single device.

4.1.6 DVBCSS.jss

Library with an implementation of the DVB-CSS (Companion Screens & Streams) protocol, adopted by HbbTV standard, to provide Inter-Device Synchronization (IDES) between a main TV and companion devices. In ImAc, this library will be also used to synchronize the playback between the involved devices.

4.1.7 SESSION_MANAGER.js

This library implements all the required functionalities for Session Management. It stores relevant information about the session, such as the metadata obtained from the Content Manager (e.g., describing the available contents, their relationships, etc.), the contents being currently presented, the available personalisation options together with the current settings, as well as the status of the session (e.g., elapsed time, duration, other active devices...).

5. Size and Performance

The chosen software architecture supports the key sizing and timing requirements, as stipulated in the Platform Specification.

P1	Production Editors
P2	Accessibility Content Manager
P3	Content Packager and Distribution
P4	Player

5.1. Production Editors

Key sizing and timing requirements for the production editors:

P1.1	Unlimited number of simultaneous web editors in remote stations (it depends only on the hardware capabilities of the server and authorized instances, not limited by software).
P1.2	Instant playback for previewing/verification, only a small buffer on the player when additional video has to be downloaded.
P1.3	Significant delay on the HMD verification playback, all the processes from the content packager have to be triggered.

5.2. Accessibility Content Manager

Key sizing and timing requirements for the accessibility content manager:

P2.1	File and database capacity only limited by hard drive size of the server, not by software.
P2.2	Subtitling file upload/download: no limitation in size (they are manageable files) and fast to transfer.
P2.3	Audio description file upload/download: size only limited by Apache configuration (they can have considerable size), and some time is required.
P2.4	Sign language file upload/download: size limited only by Apache configuration (they are big files), and some time without closing the web browser is required.
P2.5	Video file upload/download: size limited by Apache configuration (they are quite big files), and quite a lot of time without closing the web browser is required.
P2.6	Metadata viewing/editing: fast and without limitation (only authorized users).

5.3. Content Packager and Distribution

Key sizing and timing requirements for the content packager and distribution:

P3.1	Encoding: One simultaneous stream for main content.
P3.2	Encoding: One simultaneous stream for enhancement.
P3.3	Packaging: One simultaneous stream.
P3.4	Delivery: CDN- unlimited number of players
P3.5	Delivery: Broadcast TS- unlimited number of players

5.4. Player

Key sizing and timing requirements for the player and shared sessions:

P4.1	There is no reason to have multiple active players per consumer device. The software architecture and design of the ImAc player guarantee a proper presentation of the immersive and accessibility contents in a single player on current consumer devices (e.g. smartphones, tablets, HMDs...). It is true even considering the use of web-based technologies, and the limited hardware and software resources of the current consumer devices.
P4.2	The designed solutions to achieve a proper association between the involved devices and a synchronized playback in the session in local scenarios are scalable and lightweight. Thus, the number of simultaneous devices presenting ImAc media contents in local scenarios will be mostly limited by the available bandwidth (e.g. determined by the contract with the Internet Service Provider (ISP).
P4.3	The total number of involved players, in all the active distributed scenarios, will be determined by the scalability of the service provider / broadcaster resources (e.g., Content Manager, CDNs, number of servers...). A proper dimensioning of these resources, together will proper strategies to distribute/balance them will maximize the scalability in terms of number of concurrently ImAc players and sessions.

6. CONCLUSIONS

This document has provided a comprehensive architectural overview of the ImAc project, using a number of different architectural views to depict key aspects of the system. It is intended to capture and convey the significant architectural decisions, which have been made on the system based on User requirements (D2.2), and Platform Specifications (D2.3) defined in WP2.

In Chapter 2 we have described what we need to do. This is based on the user requirements gathered in D2.3. Chapter 3 explained how we are going to do it and provides a blueprint for the ImAc system. Chapter 4 details the tools we are going to use to make the platform work. Finally Chapter 5 describes the size and performance requirements in order for success.

This document also provides the basis for the development work in WP3 and WP4. Specifically it feeds into the following tasks:

- Immersive Platform
    - T3.2 Content formats and storage
    - T3.3 Content Packaging and distribution
    - T3.4 Accessibility Interface
    - T3.5 Player
    - T3.6 Integration and Testing
- Accessibility Service
    - T4.1 Subtitling Services
    - T4.2 Enhanced Audio Services
    - T4.3 Sign Language

Section 2.1 (Key Requirements), 3.1 (Scenarios) and 5 (Size and Performance) also form the key metrics for D3.6 (Integration and Testing Report), scheduled for an iterative release in both M10 and M20. During development of the ImAc project this document will be revised to reflect changes and a final version published in M16.

REFERENCES

[1] Kruchten, Philippe (1995, November). Architectural Blueprints — The “4+1” View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50.

<END OF DOCUMENT>