Deliverable

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

D3.3 Content Packaging and Distribution

Revision: 1.8

Authors: Marc Brelot, Romain Bouqueau, Rodolphe Fouquet (Motion Spell)

Delivery date: 01 - 08- 2019 (M22)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 761974		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

Abstract:

This report focuses on software modules implemented in task 3.3 "Content packaging and distribution". First, it presents an overview of the current ImAc platform architecture and details the workflow and each of its components. It also details the future evolutions of the platform regarding final iteration needs of the project, specially linked to pilot 2. Then, this report explains how those modules can be installed and managed (user manual section) and how to access the code.

REVISION HISTORY

Revision	Date	Author	Organisation	Description
0.1	28-06-2018	Marc Brelot	MSE	Template and ToC
0.2	01-07-2018	Romain Bouqueau	MSE	Adding initial content
0.3	06-08-2018	Marc Brelot	MSE	Modifying / Adding content
0.4/0.5	08-08-2018	Romain Bouqueau Rodolphe Fouquet	MSE	Modifying / Adding content
0.6	10-08-2018	Marc Brelot	MSE	Finalize a first temporary version
0.7	07-09-2018	Rodolphe Fouquet	MSE	Modifying / Adding content
1.0	08-09-2018	Marc Brelot	MSE	first complete draft for review
1.1	13-09-2018	Marc Brelot	MSE	Template adaptation Submitted version
1.2/1.3	08-10-2018	Marc Brelot	MSE	Reviewed version addressing the comments by Francesc Mas (CCMA)
1.4/1.5	11-11-2018	Marc Brelot	MSE	Reviewed version addressing the comments by Mario Montagud (I2CAT)
1.6	06-07-2019	Marc Brelot Rodolphe Fouquet	MSE	Adding chapter on future possible evolutions
1.7	29-07-2019	Marc Brelot Rodolphe Fouquet	MSE	Reviewed version addressing the comments by Mario Montagud (I2CAT)
1.8	01-08-2019	Marc Brelot Rodolphe Fouquet	MSE	Final reviewed version addressing the comments by Mario Montagud (I2CAT)

EXECUTIVE SUMMARY

The ImAc project Task T3.3 "Content packaging and distribution" defines how the media contents generated by the Production tools, described in D4.1[3], D4.2[4] and D4.3[5], are prepared and managed through the Accessible Content Manager (ACM), described in D3.2, and then packaged and delivered to the player, described in D3.5 [2].

This deliverable focuses on the software developed in that task, by describing the global architecture, the different modules and the workflow. This document aims at exposing first what has been currently developed for the first phase of the project (mostly linked with the pilot 1) and then presents the planed evolutions of architecture and modules that would fulfil the final needs of pilot 2. In parallel, the specification and implementation of the different modules strive to rely as much as possible on standards, and to provide generic APIs allowing to setup a full production chain open to scalability and extension.

CONTRIBUTORS

First Name	Last Name	Company	e-Mail
Romain	Bouqueau	Motion Spell	romain.bouqueau@gpac-licensing.com
Marc	Brelot	Motion Spell	marc.brelot@gpac-licensing.com
Rodolphe	Fouquet	Motion Spell	rodolphe.fouquet@gpac-licensing.com
Francesc	Mas	CCMA	fmas.z@ccma.cat
Mario	Montagud	i2CAT	mario.montagud@i2cat.net

CONTENTS

TABLES OF FIGURES AND TABLES

LIST OF ACRONYMS

Acronym	Description
ACM	Accessibility Content Manager
AD	Audio Description
API	Application Programming Interface
AST	Audio Subtitles
CRON	**chron**o table (time-based job scheduler)
DASH	MPEG Dynamic Adaptive Streaming over HTTP
JSON	JavaScript Object Notation
MPD	Media Presentation Description: the DASH manifest/playlist
MD5	message-digest algorithm
OTT	Over The Top
PHP	Hypertext Preprocessor
REST API	Representational State Transfer API
SFTP	Secure File Transfer Protocol
SL	Sign Language
ST	Subtitles
XML	Extensible Markup Language

1. INTRODUCTION

1.1 Purpose of this document

This deliverable focuses on software modules implemented in task 3.3 "Content packaging and distribution". First, it presents an overview of the current ImAc platform architecture and details the workflow and each of its components. It also details the future evolutions of the platform regarding final iteration needs of the project, specially linked to pilot 2. Then, this report explains how those modules can be installed and managed (user manual section) and how to have access to the code.

1.2 Scope of this document

The objective of task T3.3 is to provide a set of modules to achieve the packaging and the distribution of media contents including innovative formats used into the ImAc project, in particular immersive accessibility formats. This task is also actually linked with the task T3.2 "Accessibility Content Manager", described in deliverable D3.2 and task T3.5 "Player" described in D3.5 [2]. In this sense, the T3.3 objectives are to provide a full set of content processing services which the ACM and the player can rely on. Moreover, this task also includes the provisioning of servers on which the ACM and the Content Packaging & Distribution services are installed.

Then, in this deliverable, we will describe:

- The ImAc server-side platform as an infrastructure.
- The ingest transcoding service that provides appropriate quality for production.
- The packaging and distribution service to provide multiple qualities for adaptive delivery.

1.3 Status of this document

The previous version of the document focused on the software components and modules linked to the first iteration of the project (pilot 1). This version report the progress made towards the second iteration (pilot 2). Some aspects regarding the automated workflow are still ongoing, so they and will be refined and completed at a later stage and reported in an updated version of the delivrable.

1.4 Relation with other ImAc activities

The development of the ImAc packaging and the distribution set of modules, together with other components of the Immersive Platform and the player, is included in WP3. It is driven by the (home + professional) user requirements established in WP2, but refinements and extensions will be included, based on the insights from the evaluations in WP5, for each one of the pilot phases considered in ImAc. Improvements can also be integrated based on the results from the integration tasks in WP3 and from feedback gathered in dissemination actions (WP6). This is illustrated in Figure 1.

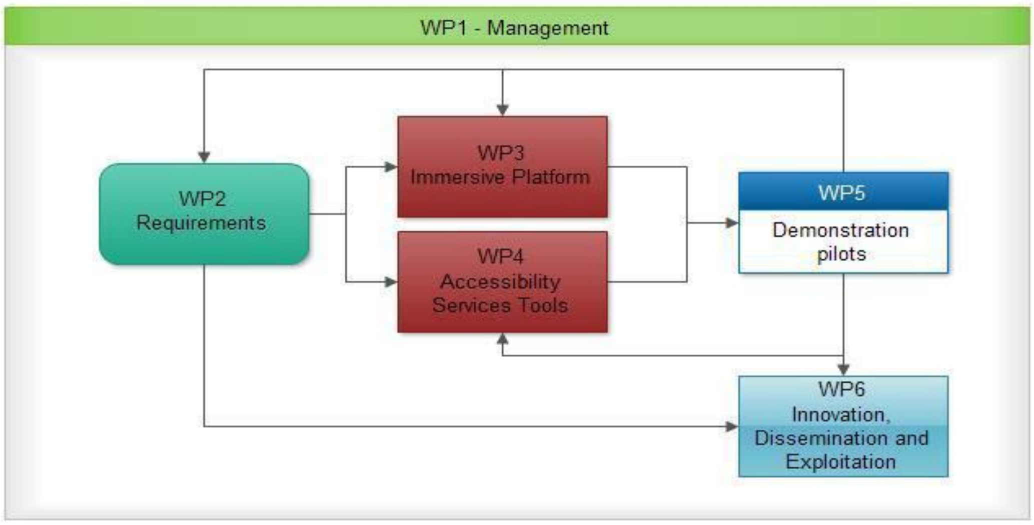

WP1 - Management
WP2 Requirements
WP3 Immersive Platform
WP4 Accessibility Services Tools
WP5 Demonstration pilots
WP6 Innovation, Dissemination and Exploitation

Figure 1: Relationships between Work Packages (WPs), and their cycles (iterations).

In addition, the PERT diagram in Figure 2 illustrates with more details the relation between T3.3 and the other ImAc activities. In particular, the set of modules provided into T3.3 will be a bridge between the Accessibility Content Manager (T3.2) and the player (T3.5). The packaging and distribution modules have been developed in line with the Architecture Design tasks reflected in D3.1.

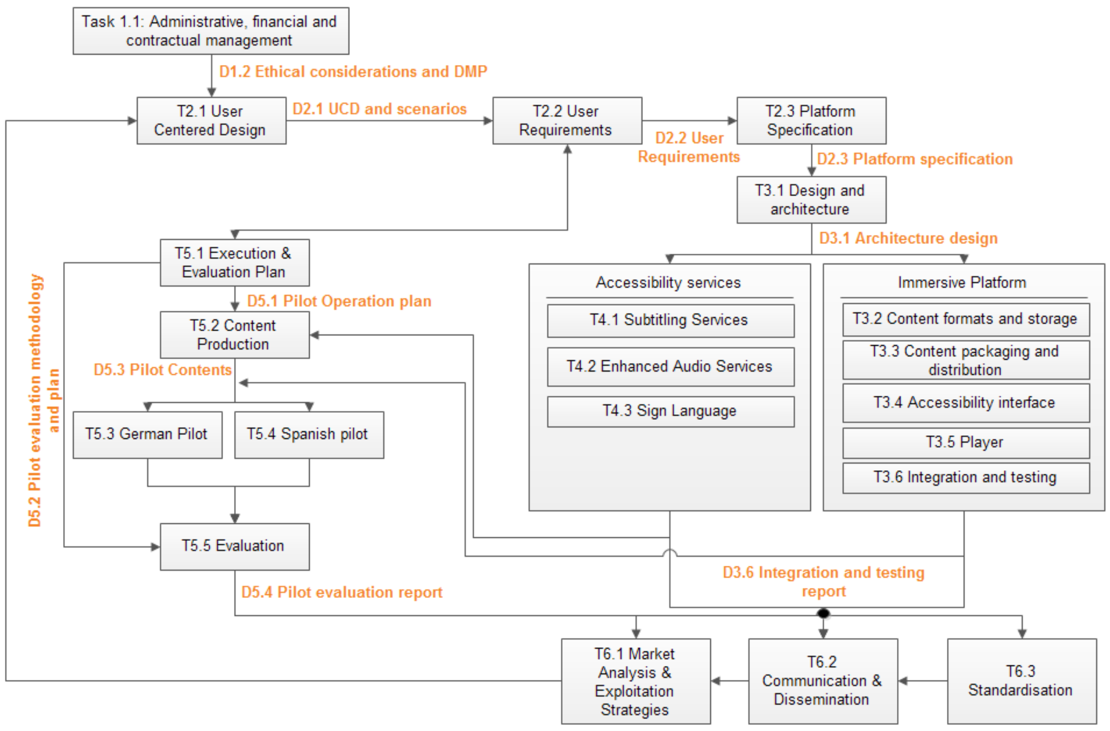

Task 1.1: Administrative, financial and contractual management
D1.2 Ethical considerations and DMP
T2.1 User Centered Design
D2.1 UCD and scenarios
T2.2 User Requirements
D2.2 User Requirements
T2.3 Platform Specification
D2.3 Platform specification
T3.1 Design and architecture
D3.1 Architecture design
T5.1 Execution & Evaluation Plan
D5.1 Pilot Operation plan
T5.2 Content Production
D5.3 Pilot Contents
T5.3 German Pilot
T5.4 Spanish pilot
T5.5 Evaluation
D5.4 Pilot evaluation report
D5.2 Pilot evaluation methodology and plan
Accessibility services
T4.1 Subtitling Services
T4.2 Enhanced Audio Services
T4.3 Sign Language
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
D3.6 Integration and testing report
T6.1 Market Analysis & Exploitation Strategies
T6.2 Communication & Dissemination
T6.3 Standardisation

Figure 2: PERT Diagram illustrating the relationship between T3.3 and other ImAc activities.

2. PLATFORM DESCRIPTION

The ImAc platform is a complete end-to-end chain with key parts: content preparation parts, server-side part and a client-side part (player). This chapter focuses on the server-side part of the platform and more precisely on the set of components to provide all services linked to the ingestion, preparation, packaging and distribution of media contents. This platform is evolving during the iteration phases of the project. The chapter 3 focuses on the current version of packaging and distribution components implemented and tested within the framework of the pilot 1. The chapter 4 presents the future evolutions of the packaging and distribution components to fulfil needs of the next iteration of pilot actions.

2.1. General architecture

The D3.1 document specifies a general architecture which is a second iteration of the architecture updated for pilot 2 with the different components as it is shown in the figure below:

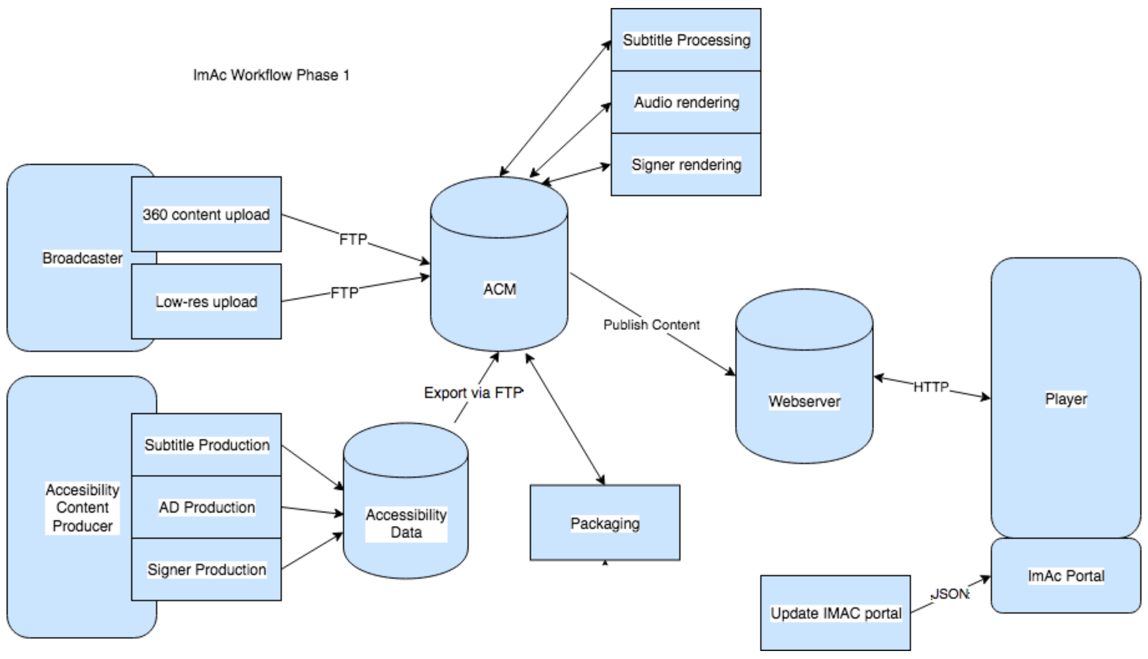


Figure 3: Deployment View Diagram - complete ImAc system

2.2. Server-side platform implementation

The schema below presents the general implementation architecture of the server-side ImAc platform:

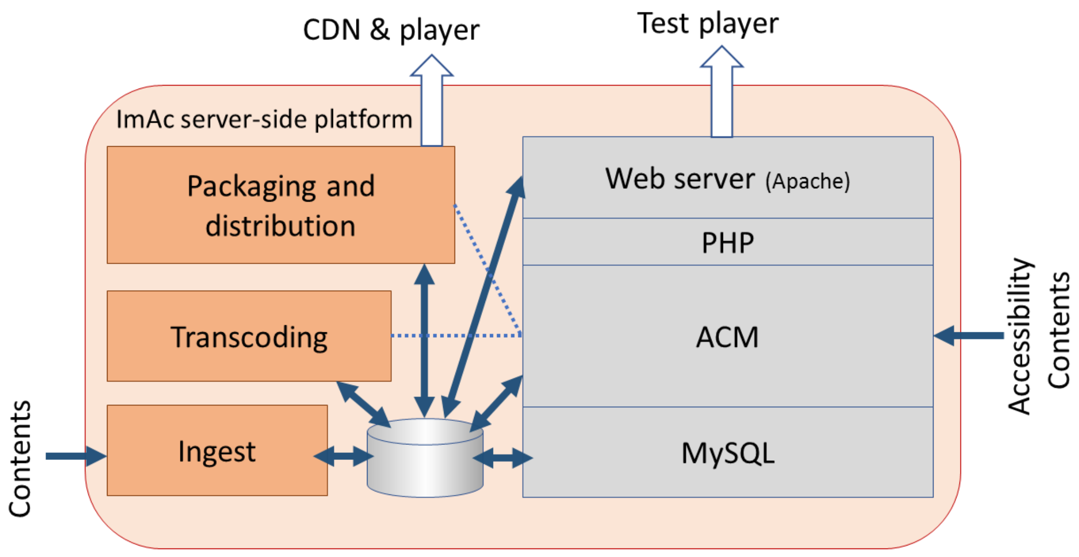


Figure 4: General platform architecture implementation

The ImAc server-side platform hosts 3 mains functional components:

- An ingest module, which allows content producers/owners to upload their contents and then to transcode them with an appropriate quality for the ACM.
- The (ACM) components, which are described in D3.2.
- The "Content packaging and distribution" component.

All components on the right (in grey) correspond to modules that work with the ACM, which are described into the deliverable D3.2. The other components (in orange) are described in this document with three main parts: ingest, transcoding, and packaging and distribution.

2.3. General workflow

In the ImAc workflow (described in D3.1), audio-visual contents coming from the producer/owner are firstly ingested into the platform and transformed into an appropriate format and/or encoded into an appropriate quality. Then, accessibility contents like subtitles authored thanks to editing tools are then linked to the corresponding contents within the ACM. The workflow can then be divided into the next steps:

1. **Ingest:** the audio-visual input contents are retrieved at a high quality level from the mastering process of production: high resolution, high bandwidth, specific master production codecs.
2. **Transcoding:** Since the editing tools need to work with low bandwidth content, the audio-visual contents are transcoded before being sent to the ACM.
3. **ACM:** The accessibility content is then linked with audio-visual contents through different tools of the ACM (like the web-based player connected to the ACM server).
4. **Distribution:** The audio-visual contents and the corresponding accessibility contents are prepared for their distribution. Qualities from the original (4K or even higher) to lower acceptable qualities (up to 720p, or lower for the SL video) are transcoded, packaged, and made available, via broadcast

3. ARCHITECTURE AND COMPONENTS DESCRIPTION (PHASE 1)

This chapter describes more in detail the architecture and the first implementation phase of the different components. This first implementation has been used for the pilot 1, but it is also used for pilot 2. However, the requirements coming with pilot 2 actions lead to improve either the global architecture and also components linked to the packaging and distribution. Those improvements are described in next chapter.

3.1. Server-side platform implementation

The schema below presents the implementation architecture of the phase 1 of the server-side ImAc platform in which a SFTP server have been chosen as the ingest:

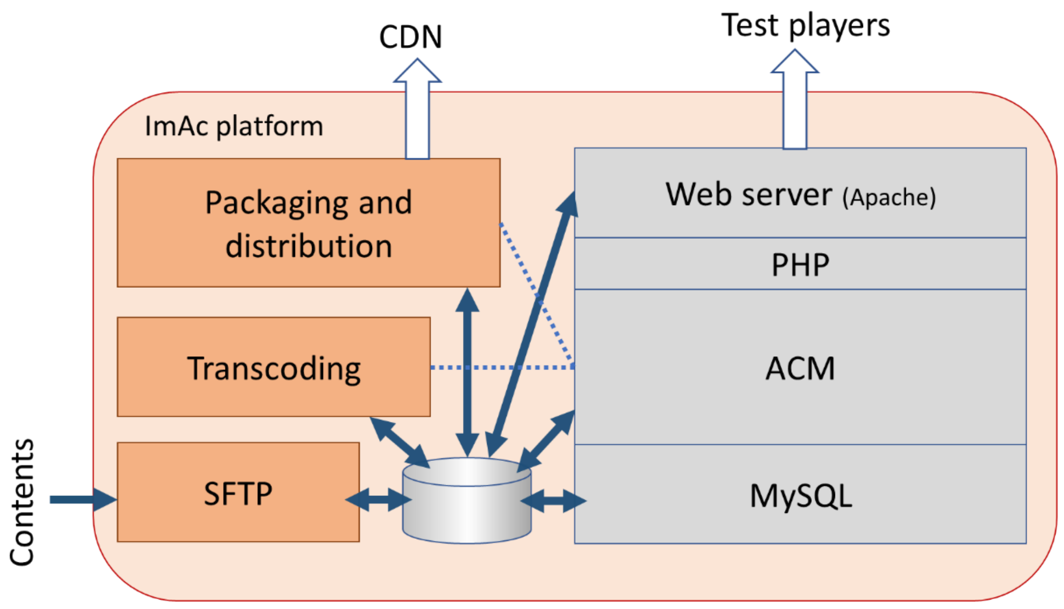


Figure 5: Platform architecture implementation for phase 1

3.2. Server-side platform workflow

To have a better understanding of the processes for contents creation and preparation, the schema in Figure 5 indicates the different sequential steps, and thus the content workflow, through the different modules of the platform. These steps are also listed below:

1. AV content is ingested through the SFTP, which is a secured FTP server.
2. AV content is transcoded for the ACM.
3. Accessibility Content is authored by using the editing tools linked to the ACM (ST, AD, SL).
4. Accessibility & AV content is packaged
5. Packaged content is then converted into DASH format (i.e. encoded in multi-qualities, segmented) and the appropriate metadata files are created (i.e., MPD, and updated list of available contents) and published on the webserver for their consumption by using the ImAc player. The ImAc player can also use the JSON written by the packager (Annex 5) to list and describe the available contents.

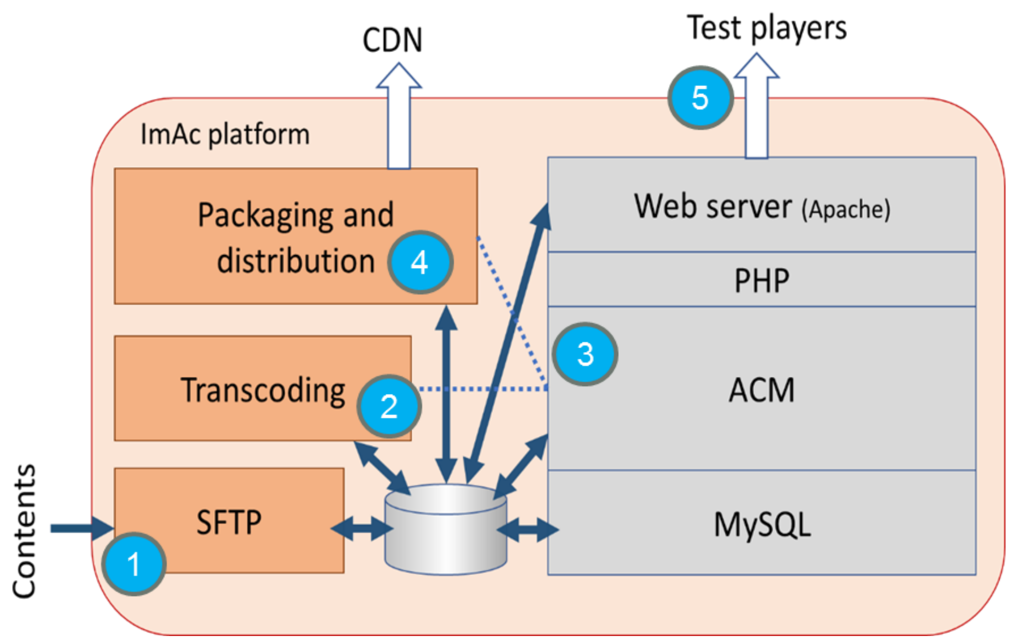


Figure 6: Content workflow through the platform

3.3. Content preparation sequence

In order to detail more the workflow and the underlying process, a sequence diagram is presented below. This diagram summarizes the content transformation steps through the different processes running onto the ImAc platform and the events sent between processes or external actors. The associated Concepts and implementation details are explained in next sections.

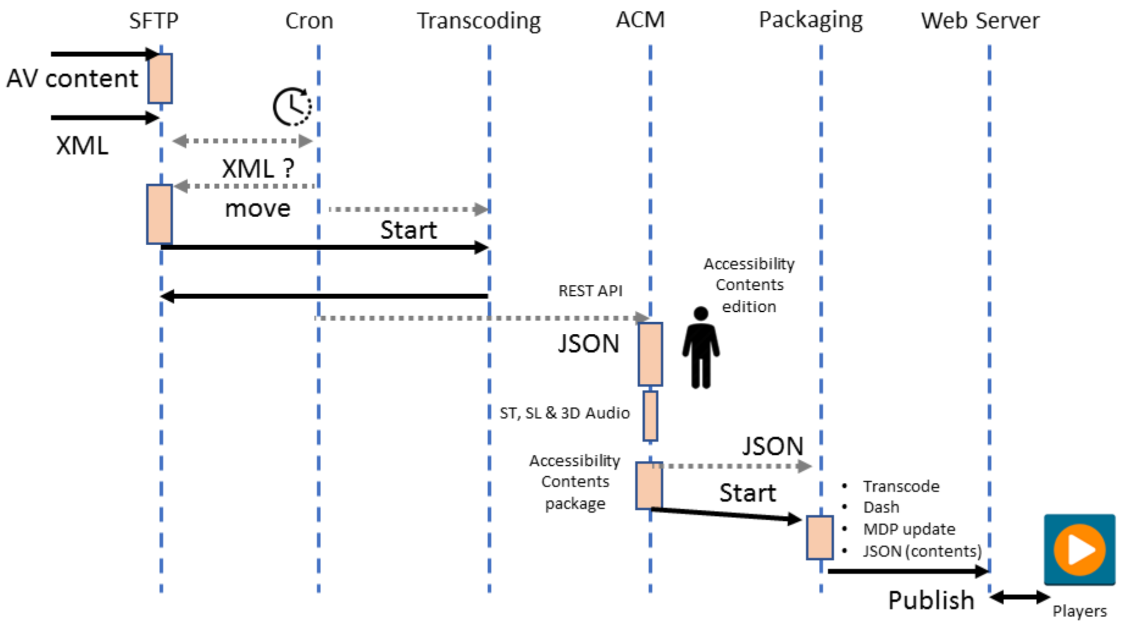


Figure 7: Sequence diagram of the content processing

3.4. Components description

In this section, we describe how the "Content packaging and distribution" components work and how they interact with the rest of the ImAc platform.

3.4.1. Ingest

The ingest module allows to upload contents from production to the ACM.

3.4.1.1. Ingest steps and metadata description file

For the ingest phase, the provider will have to upload assets onto the input folder of the SFTP server and then upload a metadata description file. For pilot 1 and 2, a metadata description has been specified to be able to expose all necessary parameters to start the transcoding process. This description file is based on XML (Extensible Markup Language), which is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. Moreover, there are XML validators like XSD.

An example of metadata description file is given in Annex 1. This file exposes parameters like file name, size, number of channels for audio, type of encoding, languages, and so on.

This module is based on CRON script (https://en.wikipedia.org/wiki/Cron). CRON is a time-based job scheduler to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals. CRON has been chosen because of its flexibility and easy update capabilities during this first phase of construction of the all end-to-end process. In the next phase (see chapter 4), CRON is likely to be replaced by an actual scheduler.

After this step, the CRON process task will do:

- Transcoding of new ingested audio-visual content to an audio & video format appropriate for the ACM.
- The ACM will generate the required metadata document to ensure consistency; this includes a human readable indexing of the content, which provides information necessary to automatize later steps in the ACM (see D3.2 for details on these steps).
- Generation of a JSON notification, which is sent to the ACM to start the task of authoring.

NOTE: JSON (https://en.wikipedia.org/wiki/JSON) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.

3.4.1.2. Transcoding module and CRON script

The transcoding module aims at processing a high quality video (ingest quality) into a lower quality video in order to provide the ACM with an appropriate file to work with. When the transcoding task is done, the module will call the ACM through a REST API with a JSON result file.

To summarize, the main steps previously described to be executed within this CRON script are:

- List the XML files in the input folder of the SFTP (along with the media files).
- Parse and validate the XML files one by one.
- Check that the uploaded files are on the disk and have the correct size provided in the XML.
    - If the check fails, the corresponding files are moved into an "error directory".
    - If the check is ok, the XML and the files it lists are moved to a "working" directory, and then the XML is then processed.
- Start the transcoding process
    - If the processing fails, an error message is output and files are moved to the "error directory" mentioned above to allow replication of the issue.
    - If the processing is ok, files are moved to the output directory.
- Call the ACM through the REST ACM API with a JSON result file (dashed line in Figure 7). An example of JSON file for this process is given in Annex 2.

NOTE: In Phase 1, low quality video with 2 audio channels has been agreed as the required format for the ACM input.

The use of a CRON script file was guided by practical reasons: the project needed a way to simulate some tools that broadcasters have at hand, but couldn't share easily and quickly with the developers. However, the current approach has drawbacks detailed in this section. The current limitations while uploading contents are:

- If the CRON job is set to occur every X minutes and a file/set of files takes longer than that time to process, the transcoding process may be sent several times, which will slow the computer even more, leading to CPU-locking the server. Therefore, the CRON frequency is quite low (every 20 minutes given that typical contents are less than 10 minutes).
- The SFTP upload is not atomic, so if the CRON job is being triggered while the XML file is being written, an error will be sent (media files are already checked from the XML).
- If the XML file is, by mistake, sent before the media assets, the script will detect missing files and send the files to the error folder.
- The XML file itself may be written by a human and then the edited file may be error prone.
- The use of a CRON job makes debugging pretty hard to do without reprocessing the files manually. Reprocessing manually is however the good practice to reproduce bugs out of the server.
- The use of the file size instead of the MD5 checksum to verify data integrity (https://en.wikipedia.org/wiki/MD5) won't check if the file is valid. It is a basic verification, but not a very efficient one. This limitation was imposed by broadcasters who need to edit the file manually and were not comfortable with checksums.

3.4.2. Low Quality transcoding

3.4.2.1. ImAc encoder

The ImAc Encoder (**imac-encoder)** takes the previously mentioned XML file as an input to define the job processes and transcodes the audiovisual contents as needed. This process involves some processing on the audio, which can be ingested as stereo, binaural or 3D spatialized, and needs to be downmixed to a format understandable to the ACM preview component.

For phase 1, imac-encoder is handled as a shell script triggering the FFmpeg transcoding application (https://www.ffmpeg.org/documentation.html). In particular, the FFmpeg command-line for generating the low quality preview for the ACM is:

```
ffmpeg -i video.mp4 -i audio.wav -vf scale=-2:720 -c:v libx264 -bf 0
-crf 22 -c:a aac output.mp4
```

This command-line will transcode the original high quality video into a new lower quality content compatible with the ACM, where:

- Video resolution is rescaled to 720p
- Video compression is H264
- Audio is stereo and coded in AAC

3.4.2.2. Notification to the ACM

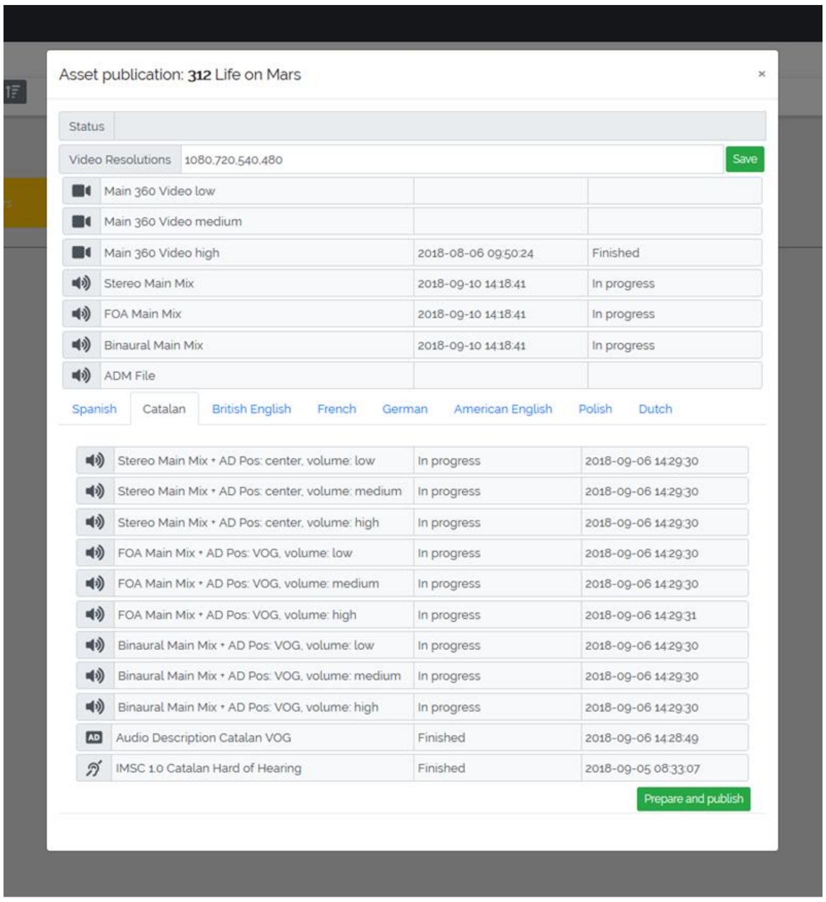
When the transcoding process is completed, the ACM is notified and it receives a JSON file like the following one:

```
{
  "pathXML": "/home/alex/projects/imac/497288-Life_On_Mars.xml",
  "pathVideoLow": "/home/alex/projects/imac/497288-Life_On_Mars-low.mp4",
  "pathVideoHigh": "/home/alex/projects/imac/497288-Life_On_Mars.mp4",
  "audiosHigh" :
       [
       {
          "path" : "/home/alex/projects/imac/497288-Life_On_Mars_1.wav"
        },
       {
          "path" : "/home/alex/projects/imac/497288-Life_On_Mars_2.wav"
        }
    ]
}
```

3.4.3. Packaging and distribution module

3.4.3.1. From ACM to the packaging

Once the ACM has received the audiovisual contents, professional users will use the different interfaces to create, add or modify accessibility contents (namely: SL, AD, ST) described in deliverables 4.1, 4.2 and 4.3. When the user finishes the preparation of these accessibility contents, he/she will have to fill a form with parameters and options, as it is shown in the snapshot below:



Asset publication: **312** Life on Mars

Status	
Video Resolutions	1080,720,540,480

Main 360 Video low		
Main 360 Video medium		
Main 360 Video high	2018-08-06 09:50:24	Finished
Stereo Main Mix	2018-09-10 14:18:41	In progress
FOA Main Mix	2018-09-10 14:18:41	In progress
Binaural Main Mix	2018-09-10 14:18:41	In progress
ADM File		

Stereo Main Mix + AD Pos: center, volume: low	In progress	2018-09-06 14:29:30
Stereo Main Mix + AD Pos: center, volume: medium	In progress	2018-09-06 14:29:30
Stereo Main Mix + AD Pos: center, volume: high	In progress	2018-09-06 14:29:30
FOA Main Mix + AD Pos: VOG, volume: low	In progress	2018-09-06 14:29:30
FOA Main Mix + AD Pos: VOG, volume: medium	In progress	2018-09-06 14:29:30
FOA Main Mix + AD Pos: VOG, volume: high	In progress	2018-09-06 14:29:31
Binaural Main Mix + AD Pos: VOG, volume: low	In progress	2018-09-06 14:29:30
Binaural Main Mix + AD Pos: VOG, volume: medium	In progress	2018-09-06 14:29:30
Binaural Main Mix + AD Pos: VOG, volume: high	In progress	2018-09-06 14:29:30
Audio Description Catalan VOG	Finished	2018-09-06 14:28:49
IMSC 1.0 Catalan Hard of Hearing	Finished	2018-09-05 08:33:07

Figure 8: Contents accessibility form

During the pilot 1, there are still some manual operations to do:

- The ST format may have to be processed by doing a TTML pre-segmentation for distribution (Such a process is described in this open recommendation: https://www.gpac-licensing.com/2016/03/21/guidelines-on-how-to-embed-ttml-in-mp4-and-dash/).
- The accessibility contents SL, AD, ST are then uploaded onto the SFTP.
- The audio will have to be processed manually, if needed.

After validation of the form, the packaging and distribution module will be triggered through a webservice and then executed with a JSON file as parameter (an example of JSON file is given in Annex 2). This JSON file describes all linked media associated to one content (audio, video, SL, AD, ST) in order to provide to the packager all information to package and adapt the whole content ready to be published.

3.4.3.2. Imac-packager

The packaging & distribution module allows to package audiovisual contents and all associated accessibility contents into a distribution format (or stream-based format, like DASH) taking into account the specificity of the different targeted platforms (PC, Android, iOS, HbbTV2.0). The distribution format, like DASH, is extended accordingly regarding the signalisation of the accessibility contents. The designed mechanisms to signalize the access services and their features via the MPD are described in D3.4 and D3.5.

The ImAc Packager (imac-packager) is aimed to be a separate tool. For the process of agile development, it is now made of a set of scripts that have been considered acceptable for Pilot 1 and 2.

The packaging and distribution module then executes several atomic operations, being the main ones:

- Parse the JSON file coming from the ACM.
- Transcode the video from the ingest format to the publication format.
- Fragment the audio and the video final encodings (according to DASH format).
- Fragment also audio description and sign language video (according to DASH format).
- Create some DASH assets (generates a MPD manifest).
- Modify the generated MPD and add TTML for the subtitles. See example in Annex 3.
- Modify the generated MPD manifest to add custom descriptors for audio (see example in Annex 4). The steps for adding discontinuous SL videos are being defined at the time of writing.
- Publish content on the HTTP server (either on the web server of the ImAc platform for testing or onto a CDN server for massive distribution).

3.5. Server setup

The ImAc server-side platform has been set up on servers provided and managed by Motion Spell. For Pilot 1 and pilot 2, the installation tasks described below have been followed:

- Setting up a Virtual Machine through Google Cloud Platform equipped with 2 CPU cores and 2 Tb of storage.
- Installation and configuration of an HTTP server (apache) and a SFTP file upload server.
- Creation of a script to be able to install and configure another server without any manual operation.
- Creation of a running task (based on CRON script) allowing to process audio-visual content input, generate JSON result files and call other components (ACM and packager).

4. ARCHITECTURE AND COMPONENTS DESCRIPTION (PHASE 2)

The platform architecture and set of components used for pilot 1 have some limitations that need to be solved in the project next phases:

- The CRON process shows many limitations: Since a CRON is triggered every X minutes, the processing latency can be up to that limit, delaying the input processing. Also, when a new CRON starts, the previous one may not have finished. If you want to have more leeway for the first CRON to finish, you may want to increase the CRON period, however, that would lead to longer delays.
- The SFTP server to upload media content is not the best for web process and specially to give the control to the ACM to lead the upload.
- Automation needs a better engine to apply rules and scripts.
- The processing time appears to be too long.
- Process many media contents (or provide a production platform to many users) appears to be complex.
- The accessibility contents processing pipeline is not yet fully automatized and still need to be improved for pilot 2.
- Some metadata file for the player need to be more flexible (like adding personalized covers for each clip).

This chapter presents how the current architecture will be improved towards more scalability and better input processing latency, which was an issue inherent to the CRON architecture of the phase 1 version.

4.1. Server-side platform implementation (phase 2)

The general server-side implementation of the new version is given below:

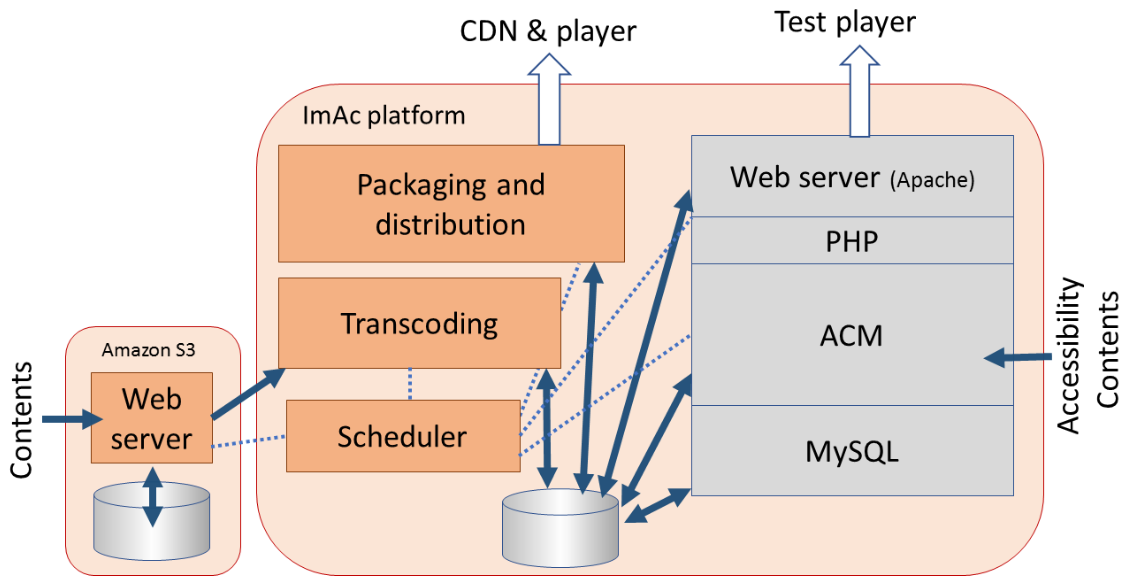


Figure 9: Platform implementation architecture for phase 2

On this new version, the following things change:

- The processing isn't done on a single server anymore. Once a content needs to be processed, the scheduler triggers a dedicated server running for that task only (detailed in 4.1.2).
- The ingestion is done through Amazon S3 instead of SFTP (detailed in 4.1.1) .

4.1.1. Better upload management

Currently, we are uploading the contents using an SFTP server. However, this method has some issues:

- We cannot automatically trigger an action after an upload, which forces us to use a CRON job, which, in turn, introduces processing latency.
- We are limited by the storage of the computer we are uploading to.
- No redundancy: if a hardware failure occurs, the files are lost.

The next version will use Amazon S3 for the following reasons:

- We can trigger an action after an upload, and this action would activate the scheduler for the uploaded content.
- Since Amazon can trigger actions, there would be no need to implement a dedicated backend for that. We plan to use S3 tools, which are widely available.
- If we ever need to implement a backend ourselves, S3 is widely supported.

4.1.2. Better scalability

As explained previously, we are currently transcoding and packaging all out videos on a single server, meaning that if multiple packaging are requested, or if multiple videos are being uploaded at the same time, we could saturate the server and make it crash or slow down dramatically. If that happens, the server's ability to respond to requests, or continue its work, could be severely hampered.

To solve that, a scalability API would be added, which would start a server dedicated to a single process. Once an encoding or packaging task arrives, it can be processed quickly, on a dedicated hardware, without risking slowdowns on other tasks.

4.1.3. Encoding Improvements

The encoding process will be improved to fix a few playback issues that have been encountered by the web player (stutters and slow playback). The encoding has been town down to the H264 main profile.

This will help the browser keep a steady playback, since it has to decode two streams and composite them into the final 360 scene.

4.1.4. Discontinuous Signer Language video and position

Currently, we are displaying the signer language video as a single continuous video. However, we do not always need to display it, since parts of the video do not have a corresponding signer language video associated to it.

Hence, the SL video needs to be chunked, and each chunk may need its own position in the immersive video.

To solve that issue, the SL video will split using the metadata sent by the ACM (to be defined), then each segment will need to be transcoded with a 2 second fixed GOP, and finally, each segment will be "DASHed" into its own period to be inserted in the MPD.

As for the position, it will be an attribute of the corresponding period.

5. HOW TO

This section aims at providing practical information for two main purposes. First, it provides the URLs and credentials to allow the professional users to access the SFTP server. Second, it describes how to access, install and use the software.

5.1. How to access the SFTP server

The SFTP server is a secured FTP server. Many applications (like Filezilla) can be used to get connected to it thanks to those information:

- ImAc Server: imac.gpac-licensing.com
  - User: imac
  - Pwd: jyfRwHyqZIDy7B
- Direct command line:
  sftp://imac:jyfRwHyqZIDy7B@imac.gpac-licensing.com

5.2. How to access the Code

To have access to all code, the ImAc server can be accessed through ssh with the same user/pwd than the SFTP access.
The code of the cron job can be also accessed from here:
https://github.com/RodolpheFouquet/ImaCron
Moreover, the code for the packaging rest API is accessible from here: https://github.com/RodolpheFouquet/Imackager

5.3. How to install the content packaging and distribution module

This part explains the different steps to follow in order to install each sub-module needed for the phase 1 of the ImAc platform. The different improvements for pilot 2 will be added in an updated version of the deliverable provided at a later stage.

5.3.1. How to install the Cron script

- Download the script main.py from https://github.com/RodolpheFouquet/ImaCron
- Install pip for python3, on Debian or Ubuntu: sudo apt-get install python3-pip
- pip install colorama
- pip install xmlschema
- edit the crontan using crontab -e
- add `*/20 * * * * /home/imac/ImacCron/main.py -i /home/imac/ftp/input -o /home/imac/ftp/output -w /home/imac/ftp/working -e /home/imac/ftp/error >> /home/imac/ftp/imacron_logs/`/bin/date +\%Y-\%m-\%d.\%H:\%M:\%S`.log 2>&1` at the end of the file
- You can tweak the input/output/error/working directories if your assets are in a different place
- You can change the imacron_logs directory if you want your logs to be written somewhere else
- You can tweak the cron interval, check out this website for help https://crontab.guru/
- Do not forget to set the correct path to the main.py file if you have copied it to a different

place

5.3.2. How to install the packager

- Download imackager.py from here https://github.com/RodolpheFouquet/Imackager/blob/master/imackager.py
- Install flash by typing “sudo pip3 install Flask”
- Run the server by typing FLASK_APP=imackager.py python3 -m flask run
- The server will be available on the port 5000
- If you desire to run it as a service, you can run it as a systemd service by isolating FLASK_APP as an environment variable

If you setup a `systemd` service, your configuration file should look like this:

```
[Unit]
Description=Imackager
After=network.target
[Service]
User=www-data
Environment=”FLASK_APP=/path/to/imackager.py”
ExecStart=/usr/bin/python3 -m flask
[Install]
WantedBy=buildbot-master.service
```

6. USER MANUAL

The different components linked to the Task 3.3 “Packaging and Distribution” are supposed be automatically executed from the ACM (or triggered from SFTP repository); manual steps are guided from the ACM as this evolves quickly.

This document constitutes the base of the user manual as well as for the installation of the different modules than for their usage.

ANNEX 1: PRODUCTION XML METADATA (AND XSD FOR VALIDATION)

2 Examples of XML contents description file that will be processed by the transcoding module:

```
<?xml version="1.0" encoding="UTF-8"?>
<content acm_virtual_folder="documentary" lang="en_GB" programmeID="497288"
title="opera">
    <inputs>
        <video>
            <file>programmeID_name.mp4</file>
            <size>36522200</size>
            <!-- Size of file in Bytes (Mandatory) -->
        </video>
        <audio>
            <file>programmeID_name.mp4</file>
            <!-- Audio is muxed in mp4 file, is the-->
            <channels>4</channels>
            <!-- Number of channels (supported: 2 for stereo, 4 for ambisonic,
object based needs further discussion) (Mandatory)-->
            <format>ambisonic</format>
            <!-- ambisonic or stereo. Could link to an external file when not
muxed with video (Mandatory)-->
            <lang>ca</lang>
            <!-- Audio language (Mandatory)-->
        </audio>
        <audio>
            <lang>ca</lang>
            <!-- Audio language (Mandatory)-->
            <file>programmeID_ name_stereo.wav</file>
            <!-- Example when audio is not muxed, or there is an additional
audio track. Is the _stereo in the name necessary? -->
            <format>stereo</format>
            <!-- Audio format (Mandatory)-->
            <channels>2</channels>
            <!-- num of channels (Mandatory)-->
            <size>787443</size>
            <!-- Size is optional. Must be provided when audio is in separate
file -->
        </audio>
    </inputs>
    <output>
        <!-- Do we need this? -->
        <targetFolder>./outputs/opera</targetFolder>
    </output>
</content>
```

The XSD allows validation of a XML specific document.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="file" type="xs:string"/>
  <xs:element name="size" type="xs:int"/>
  <xs:element name="channels" type="xs:byte"/>
  <xs:element name="format" type="xs:string"/>
  <xs:element name="lang" type="xs:string"/>
  <xs:element name="video">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="file"/>
        <xs:element ref="size"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="audio">
    <xs:complexType>
```

```
      <xs:choice maxOccurs="unbounded" minOccurs="0">
        <xs:element ref="file">
          <xs:annotation>
            <xs:documentation>Audio language (Mandatory)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="channels">
          <xs:annotation>
            <xs:documentation>Audio is muxed in mp4 file, is the Audio format
(Mandatory)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="format">
          <xs:annotation>
            <xs:documentation>Number of channels (supported: 2 for stereo, 4
for ambisonic, object based needs further discussion) (Mandatory) Example when
audio is not muxed, or there is an additional audio track. Is the _stereo in
the name necessary?</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="lang">
          <xs:annotation>
            <xs:documentation>ambisonic or stereo. Could link to an external
file when not muxed with video (Mandatory)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="size">
          <xs:annotation>
            <xs:documentation>num of channels (Mandatory)</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="targetFolder" type="xs:string"/>
  <xs:element name="inputs">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="video">
          <xs:annotation>
            <xs:documentation>Size of file in Bytes
(Mandatory)</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="audio" maxOccurs="unbounded" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Audio language (Mandatory) Size is optional.
Must be provided when audio is in separate file</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="output">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="targetFolder">
          <xs:annotation>
            <xs:documentation>Do we need this?</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="content">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="inputs"/>
        <xs:element ref="output"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="acm_virtual_folder"/>
      <xs:attribute type="xs:string" name="lang"/>
      <xs:attribute type="xs:int" name="programmeID"/>
      <xs:attribute type="xs:string" name="title"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

ANNEX 2: ACM NOTIFICATION TO THE PACKAGER

Example of JSON coming out of the ACM to be sent to the packaging and distribution module.

```
{
  "assetId": 302,
  "programName": "Life On Mars",
  "furtherProgramInformationNeeded?": "MSE please add",
  "files": {
    "mainVideo": {
              "url": "/home/alex/projects/imac/497288-Life_On_Mars.mp4",
              "urn:mpeg:dash:role:2011": "main",
              "furtherVideoInformationNeeded?": "MSE please add"
    },
    "signer":{
      "tbd": "???"
    },
    "audio":[
      {
        "url": "/home/alex/projects/imac/output/497288-Life_On_Mars-binaural-
main.aac",
        "language": "ca_ES",
            "description": "Binaural Main Mix ca",
            "audioFormat": "binaural",
            "ADposition": "",
            "ADgain": "",
            "containsAD": "0",
            "urn:mpeg:dash:role:2011": "main"
      },
      {
        "url": "/home/alex/projects/imac/output/497288-Life_On_Mars-FOA-
main.aac",
        "language": "ca_ES",
            "description": "FOA Main Mix ca",
            "audioFormat": "FOA",
            "ADposition": "",
            "ADgain": "",
            "containsAD": "0",
            "urn:mpeg:dash:role:2011": "main"
             },
             {
               "url": "/home/alex/projects/imac/output/497288-Life_On_Mars-AD-
VOG-medium.aac",
            "language": "ca_ES",
            "description": "Binaural AD VOG medium ca",
            "audioFormat": "binaural",
            "ADposition": "VOG",
            "ADgain": "medium",
            "containsAD": "1",
            "urn:mpeg:dash:role:2011": "alternate"
             },
             {
                   "etc.": "etc."
             },
             {
        "url": "/home/alex/projects/imac/output/497288-Life_On_Mars-binaural-
main-en.aac",
        "language": "en_UK",
            "description": "Binaural Main Mix en",
            "audioFormat": "binaural",
            "ADposition": "",
            "ADgain": "",
            "containsAD": "0",
            "urn:mpeg:dash:role:2011": "main"
      },
      ],
    "subtitle":[
       {
         "url": "/home/alex/projects/imac/output/497288-Life_On_Mars-sub-
ca.ttml",
         "language": "ca_ES",
         "urn:mpeg:dash:role:2011": "caption"
       },
       {
         "url": "/home/alex/projects/imac/output/497288-Life_On_Mars-sub-
en.ttml",
         "language": "en_UK",
         "urn:mpeg:dash:role:2011": "caption"
       }
       ]
  }
}
```

ANNEX 3: MPD TTML CUSTOMIZATION

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mpeg:dash:schema:mpd:2011"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"
profiles="urn:mpeg:dash:profile:isoff-live:2011,urn:com:dashif:dash264"
maxSegmentDuration="PT2S" minBufferTime="PT2S" type="static"
mediaPresentationDuration="PT1M">
   <ProgramInformation>
      <Title>Media Presentation Description by MobiTV. Powered by MDL
Team@Sweden.</Title>
   </ProgramInformation>
   <Period id="precambrian" start="PT0S">
      <AdaptationSet contentType="audio" mimeType="audio/mp4" lang="eng"
segmentAlignment="true" startWithSAP="1">
         <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
         <SegmentTemplate startNumber="1"
initialization="$RepresentationID$/init.mp4" duration="2"
media="$RepresentationID$/$Number$.m4s"/>
         <Representation id="A48" codecs="mp4a.40.2" bandwidth="48000"
audioSamplingRate="48000">
            <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011"
value="2"/>
         </Representation>
      </AdaptationSet>
      <AdaptationSet contentType="video" mimeType="video/mp4"
segmentAlignment="true" startWithSAP="1" par="16:9" minWidth="640"
maxWidth="640" minHeight="360" maxHeight="360" maxFrameRate="60/2">
         <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
         <SegmentTemplate startNumber="1"
initialization="$RepresentationID$/init.mp4" duration="2"
media="$RepresentationID$/$Number$.m4s"/>
         <Representation id="V300" codecs="avc1.64001e" bandwidth="300000"
width="640" height="360" frameRate="30" sar="1:1"/>
      </AdaptationSet>
      <AdaptationSet contentType="text" mimeType="application/ttml+xml"
segmentAlignment="true" lang="eng">
         <Role schemeIdUri="urn:mpeg:dash:role:2011" value="subtitle"/>
         <Representation id="xml_eng" bandwidth="1000">
             <BaseURL>sub_eng_short.xml</BaseURL>
         </Representation>
      </AdaptationSet>
      <AdaptationSet contentType="text" mimeType="application/ttml+xml"
segmentAlignment="true" lang="swe">
         <Role schemeIdUri="urn:mpeg:dash:role:2011" value="subtitle"/>
         <Representation id="xml_swe" bandwidth="1000">
             <BaseURL>sub_swe_short.xml</BaseURL>
         </Representation>
      </AdaptationSet>
   </Period>
</MPD>
```

ANNEX 4: MPD AUDIO CUSTOM DESCRIPTORS

```
<AdaptationSet contentType="audio" lang="en" …>
   ...
   <Representation audioSamplingRate="48000" bandwidth="260979" id="r1"
imac:ad-mode="VoiceOfGod" imac:ad-gain="2">
      <BaseURL>programName_AD_VoG_G2_binaural_seg.aac</BaseURL>
      <SegmentBase indexRange="0-2000"/>
   </Representation>
   <Representation audioSamplingRate="48000" bandwidth="260979" id="r2"
imac:ad-mode="Friend" imac:ad-gain="1">
      <BaseURL>programName_AD_Friens_G1_binaural_seg.aac</BaseURL>
      <SegmentBase indexRange="0-2000"/>
   </Representation>
   ...
</AdaptationSet>
```

ANNEX 5: JSON DATABASE USED BY THE PLAYER

```
{
  "title": "ImAc",
  "contents": [
  {
    "name": "Liceu Opera Piece 1",
    "poster": "./img/opera1_cover.png",
    "thumbnail": "img/opera1.png",
    "description": "Performance of the “Roméo et Juliette” opera
recorded at the Gran Teatre del Liceu Opera House (Barcelona). Piece
1.",
    "descriptionArray": [
    {
      "de": "Aufführung der Oper “Roméo et Juliette”, aufgenommen im
Gran Teatre del Liceu Opera House (Barcelona). Teil 1.",
      "ca": "Representació de l'òpera “Romeo i Julieta” gravada al Gran
Teatre del Liceu (Barcelona). Peça 1.",
      "es": "Representación de la ópera “Romeo y Julieta” grabada en el
Gran Teatro del Liceu (Barcelona). Pieza 1.",
      "en": "Performance of the “Roméo et Juliette” opera recorded at
the Gran Teatre del Liceu Opera House (Barcelona). Piece 1."
    }],
    "acces": [
    {
      "ST": ["en", "es", "de", "ca"],
      "SL": ["en", "es", "de", "ca"],
      "AD": ["en", "es", "de", "ca"],
      "AST": ["en", "es", "de", "ca"]
    }],
    "language": "French",
    "duration": "8:28",
    "url":"https://imac.gpac-
licensing.com/imac_content/pilot_1/OPERALICEU_01/stream.mpd",
    "audioChannels": 4,
    "subtitles": [{}]
  },
  {
    "name": "Liceu Opera Piece 2",
    "poster": "./img/covers/opera2_cover.png",
    "thumbnail": "img/opera2.PNG",
    "description": "Performance of the “Roméo et Juliette” opera
recorded at the Gran Teatre del Liceu Opera House (Barcelona). Piece
2.",
    "descriptionArray": [
    {
      "de": "Aufführung der Oper “Roméo et Juliette”, aufgenommen im
Gran Teatre del Liceu Opera House (Barcelona). Teil 2.",
      "ca": "Representació de l'òpera “Romeo i Julieta” gravada al Gran
Teatre del Liceu (Barcelona). Peça 2.",
      "es": "Representación de la ópera “Romeo y Julieta” grabada en el
Gran Teatro del Liceu (Barcelona). Pieza 2.",
      "en": "Performance of the “Roméo et Juliette” opera recorded at
the Gran Teatre del Liceu Opera House (Barcelona). Piece 2."
    }],
    "acces": [
    {
      "ST": ["de", "ca"]
    }],
    "language": "French",
    "duration": "8:15",
    "url":"https://imac.gpac-
licensing.com/imac_content/pilot_1/OPERALICEU_02/stream.mpd",
    "audioChannels": 4,
    "subtitles": [
    {
      "de":"https://imac.gpac-
licensing.com/imac_content/pilot_1/OPERALICEU_02/CCMA_OPERALICEU02_sub
titles_DE.xml",
      "ca":"https://imac.gpac-
licensing.com/imac_content/pilot_1/OPERALICEU_02/CCMA_OPERALICEU02_sub
titles_CAT.xml"
    },
    {
      "de":"https://imac.gpac-
licensing.com/imac_content/pilot_1/OPERALICEU_02/CCMA_OPERALICEU02_sub
titles_DE.xml",
      "ca":"https://imac.gpac-
licensing.com/imac_content/pilot_1/OPERALICEU_02/CCMA_OPERALICEU02_sub
titles_CAT.xml"
    }],
    "signer": [{}]
  }]
}
```

REFERENCES

[1] D3.2 : Accessibility Content Manager: see the ImAc project web site at the Work Package 3 section : http://www.imac-project.eu/documentation/deliverables/

[2] D3.5 : Player

[3] D4.1: Subtitle Production Tools : see the ImAc project web site at the Work Package 4 section: http://www.imac-project.eu/documentation/deliverables/

[4] D4.2: Audio Production Tools

[5] D4.3: Sign Language Editor

<END OF DOCUMENT>