Deliverable

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

D3.3 Content Packaging and Distribution

Revision: 1.5

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

Delivery date: 15 - 11- 2018

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 aims at focusing on the task T3.3 "Content packaging and distribution" software modules by describing the different modules and their status. It presents first an overview of the ImAc platform implementation architecture. It details also the workflow and the different service layers provided by the "Content packaging and distribution" components. Then, in the user manual section, it explains how those services have been installed, how to have access to the code and how to manage them.

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	08-08-2018	Romain Bouqueau	MSE	Modifying / Adding content
0.5	09-08-2018	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)

EXECUTIVE SUMMARY

The ImAc project Task T3.3 defines how the data generated by the Production modules (described within the Accessible Content Manager (ACM) in D3.2 in section Reference Page 24) is packaged and delivered to the end-user's players.

This deliverable focuses on the task T3.3 "Content packaging and distribution" software modules, by describing the different modules, their status and the current workflow. It presents first an overview of the ImAc platform implementation architecture. It also details the different service layers provided by the "Content packaging and distribution" components. Then, in the user manual section, it explains how those services have been installed, how to have access to the code and how to manage them.

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 ACRONYM

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

1 INTRODUCTION

1.1 Purpose of this document

This deliverable focuses on the task T3.3 “Content packaging and distribution” software modules, by describing the different modules, their status and the current workflow.

This document presents first an overview of the ImAc platform implementation architecture. It also details the different service layers provided by the “Content packaging and distribution” components. Then, in the user manual part, it explains how those services have been installed, how to have access to the code and how to manage them.

1.2 Scope of this document

The objective of task T3.3 is to provide a packaging and distribution modules which allow to deal with innovative formats used into the ImAc project, in particular accessibility formats. This task is also actually linked with the task T3.2 “Accessibility Content Manager”, described into deliverable D3.2 (see reference section page 24). In this sense, the T3.3 objectives are to provide a full set of content processing services which the ACM 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.

1.3 Status of this document

This document will be updated gradually during the project. This version focuses on the software components/modules and the infrastructure for pilot 1.

2 PLATFORM DESCRIPTION

This chapter aims at presenting the first version of the ImAc server-side platform, focusing on the production and publishing components (the client-side components are described in D3.5). This platform will evolve during the project and further updates will be explained in a next version of this document. The version 1 of the platform corresponds to the version used for the pilot 1.

2.1 General description

The ImAc server-side platform has been setting up on servers provided and managed by Motion Spell. The platform hosts 3 mains functional components:

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

2.2 Server-side platform implementation (version 1)

The schema below presents the implementation architecture of the version 1 of the server-side ImAc platform:

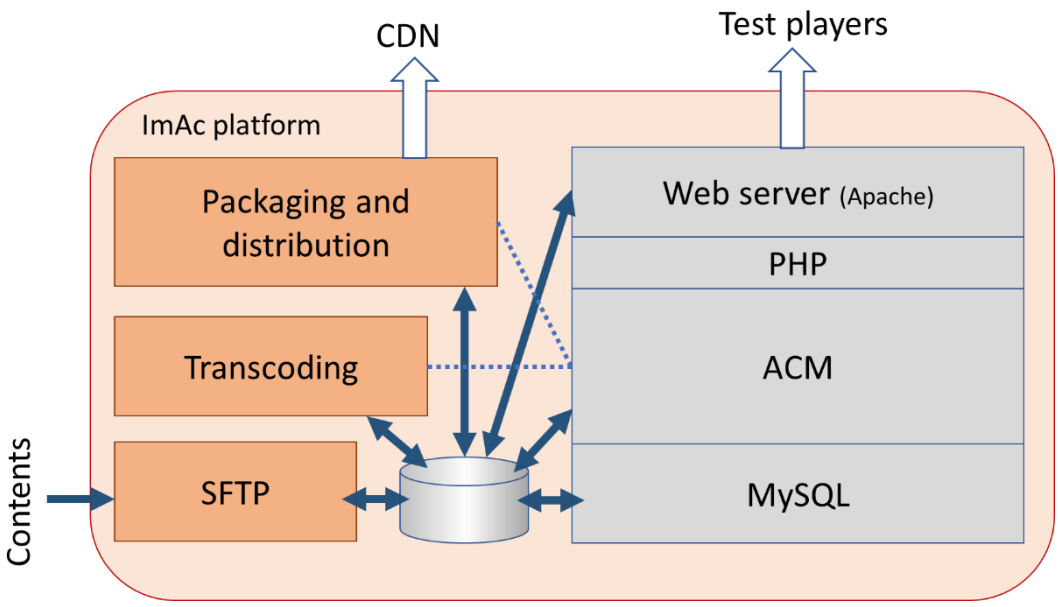


Figure 1: Platform implementation architecture for pilot 1

All components on the right (in gray) 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, packaging and distribution.

2.3 Workflow

In the ImAc workflow, audio-visual contents coming from producer/owner are firstly ingested into the platform and transformed into an appropriate format. Then, accessibility contents like subtitles are authored thanks to the ACM and linked to the corresponding contents. This workflow (out of the scope of this document) implies that the components described in D3.2 and D3.3 deal with three levels of audio-visual content quality:

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. **ACM:** the ACM needs to work with low bandwidth content, since the contents have to be previewed into a web-player connected to the ACM server.
3. **Distribution:** The audio-visual contents and the corresponding accessibility contents are prepared for their distribution in multiple qualities and formats, via broadcast and/or OTT networks**.**

To have a better understanding of the processes for contents creation and preparation, the schema in Figure 2 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
2. AV content is transcoded for the ACM
3. Accessibility Content is authored with 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 the players

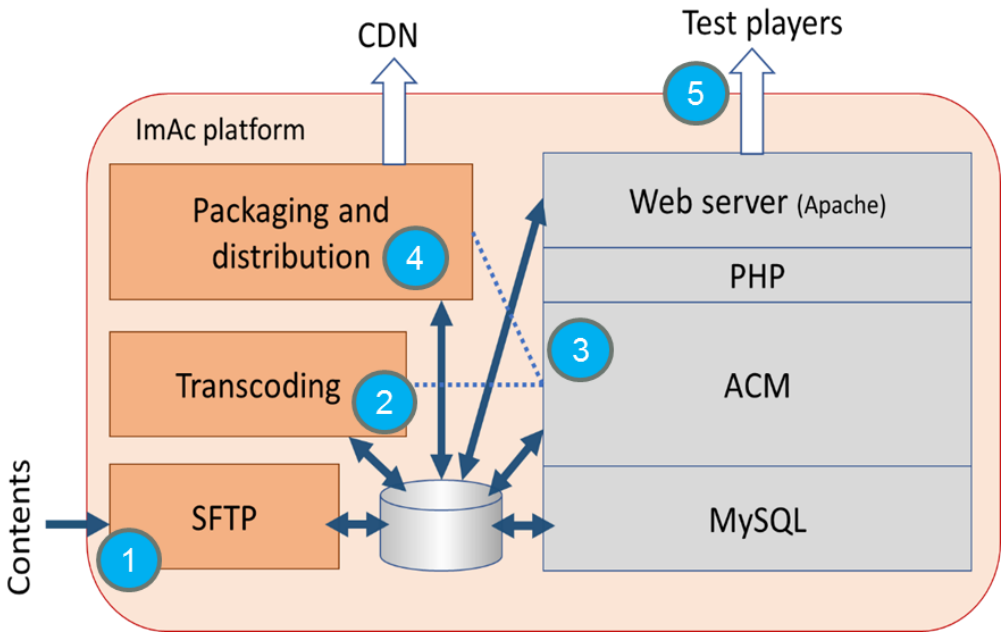


Figure 2: Content workflow through the platform

2.4 Sequence diagram

In order to detail more the workflow and the underlying process, a sequence diagram is presented below. This diagram allows to follow the content transformation through the different processes running onto the platform and the events sent between processes or external actors (Concepts and implementation details are better explained in section 3).

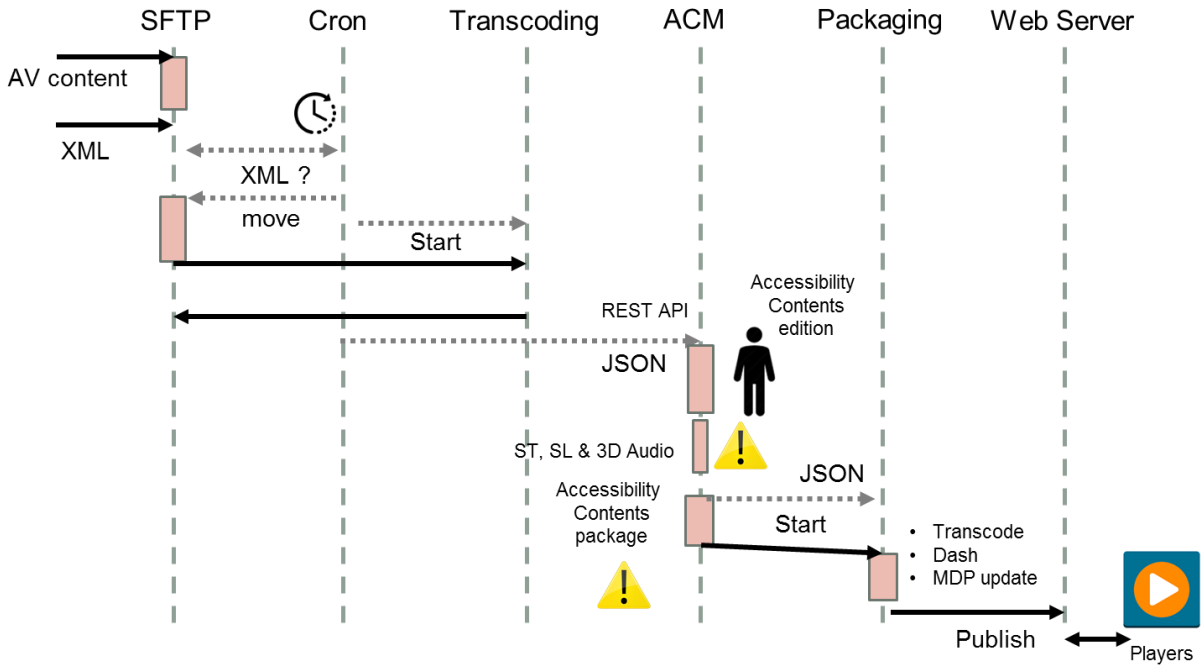


Figure 3: Sequence diagram of the content processing

In this diagram, a step is not described during publishing phase: Updating of the available contents (JSON file) when DASHing new contents.

In the final platform, all the content is supposed to be processed automatically, except for the ACM phase, where the accessibility contents have to be (co-)authored.

For pilot 1, there are 2 steps (warning icons in figure 3) that still need to be completed manually. The first one concerns audio content, where some metadata has to be added. The second one is linked to the pre-segmentation of the caption content. These steps are planned to be automatized by M18.

3 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.

3.1 Server setup

For Pilot 1, the installation tasks described below have been followed:

- Setting up a Virtual Machine through Google Cloud Platform equipped with 2 CPU cores and enough storage.
- Installation and configuration of an HTTP server 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 audiovisual content input, generate JSON result files and call other components (ACM...) .

3.2 Ingest

The ingest module allows to upload contents from the production to the accessibility infrastructure.

3.2.1 Ingest steps and the 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, a metadata description has been specified to be able to expose all necessary parameters to start the transcoding process. This description 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, an XML can be easily validated with an 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.

For pilot 1, 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 (Pilot 2 and over), CRON is likely to be replaced by an actual scheduler.

After this step, the CRON process task will do:

- A transcoding of new ingested audiovisual content to an audio & video format appropriate for the ACM module.
- 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).
- A generation of a JSON notification which is sent to the ACM to start the task of authoring.

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

3.2.2 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 a task of transcoding 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
- Call the ACM through the REST ACM API with a JSON result file (DASHed line in Figure 1). An example of JSON file is given in Annex 2.

Notice: 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. Please note that we plan to overcome these drawbacks either by plugging on a real production system or by upgrading the Cron script with a scheduling system. The Current limitations while uploading contents:

- 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 very error prone. It is planned to provide a graphically-guided tool to edit the XMLs.
- The use of a cronjob 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 (see note above).

3.3 Low Quality transcoding

3.3.1 ImAc encoder

The ImAc Encoder (**imac-encoder)** is a piece of software of the ImAc platform developed by Motion Spell. It 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 Web Browser preview component.

For pilot 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 of 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 ACM, where:

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

3.3.2 Notification to the ACM

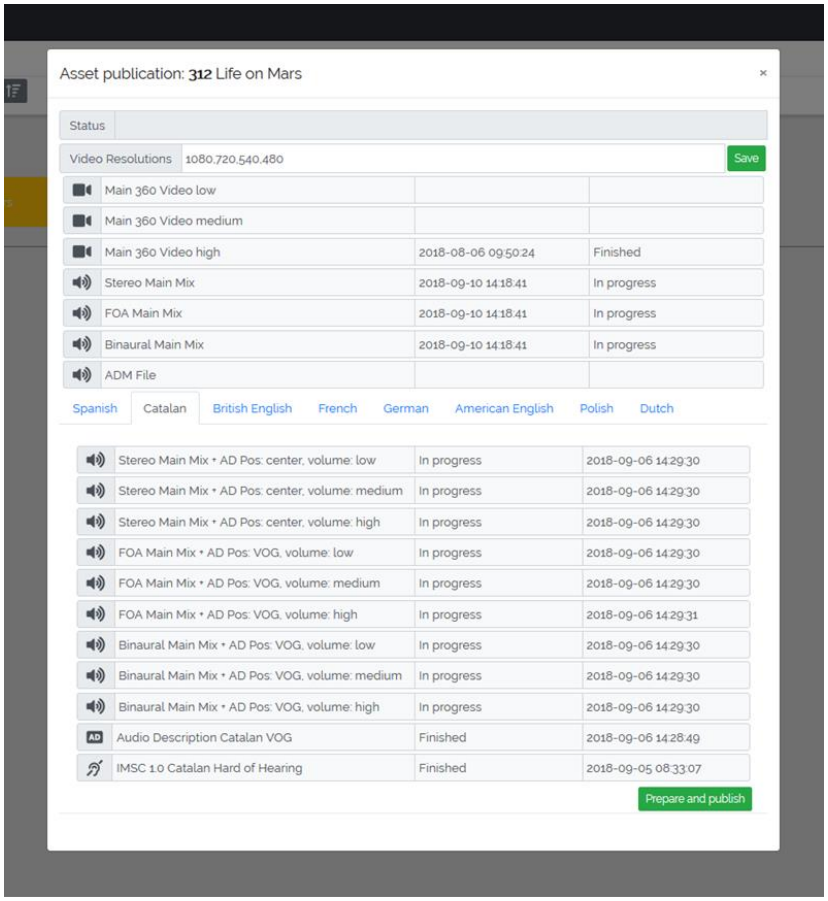
When the transcoding process is completed, the ACM is notified and it receives a JSON file like the following one:

```
{
  "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 Packaging and distribution module

3.4.1 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). 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:



In the framework of pilot 1, there are still some manual operations to do. But some improvements will be done later on during the project when all the steps will be processed automatically:

- 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.

3.4.2 Imac-packager

The packaging & distribution module allows to package audiovisual contents and associated all 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 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. These limitations have been considered as acceptable for Pilot 1.

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)
- 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).

4 HOW TO

This part 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.

4.1 How to access the SFTP server

The SFTP server is an 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

4.2 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

4.3 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 version 1 of the ImAc platform.

4.3.1 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

4.3.2 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
```

5 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 the use of those modules.

6 ANNEXES

6.1 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>
```

6.2 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"
      }
      ]
  }
}
```

6.3 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>
```

6.4 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>
```

7 REFERENCES

[1] D3.2 : Accessibility Content Manager (version 0.4) : PUBLIC version IMAC-D3.2_Accessibility Content Manager_v.04_public.pdf

https://drive.google.com/open?id=1ToqtTQ1pwLFI-0yysR3EZ03L0Q6KSDWk

<END OF DOCUMENT>