Deliverable

Project Acronym:	**ImmersiaTV**
Grant Agreement number:	688619
Project Title:	*Immersive Experiences around TV, an integrated toolset for the production and distribution of immersive and interactive content across devices.*

D3.5 Distribution and Reception

Revision: 0.2

Authors:

David Gomez (i2CAT), Einar Meyerson (i2CAT)

Delivery date: M8

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 688619

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

Abstract: This deliverable describes the hardware and software components inside the distribution and reception pipeline of media content delivered for Pilot 1.

REVISION HISTORY

Revision	Date	Author	Organisation	Description
0.1	09/01/2017	David Gomez, Einar Meyerson	i2CAT	Software description
0.2	11/01/2017	Joan Llobera	i2CAT	Document review

CONTRIBUTORS

First Name	Last Name	Company	e-Mail
Joan	Llobera	i2CAT	Joan.llobera@i2cat.net

CONTENTS

TABLE OF FIGURES

LIST OF ACRONYMS

Acronym	Description
API	Application Programming Interface
BW	Bandwidth
DDoS	Distributed denial-of-service
GUB	GStreamer - Unity Bridge
HMD	Head Mounted Display
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MPD	Media Presentation Descriptor
MPEG-DASH	Moving Picture Expert Group - Dynamic Adaptive Streaming over HTTP
REST	Representation State Transfer
URI	Uniform Resource Identifier
VOD	Video on Demand
XML	Extensible Markup Language

1. INTRODUCTION

ImmersiaTV offers a whole distribution and reception pipeline of MPEG-DASH (Moving Picture Expert Group - Dynamic Adaptive Streaming over HTTP) content. Dynamic Adaptive Streaming over HTTP (DASH) [1] specifies a XML (MPD) and binary formats that enable delivery of media content from standard HTTP servers to HTTP clients.

This pipeline is implemented as a set of modules linked and built inside a virtualization tool (Docker), with a web application to set up, transcode, publish and distribute the MPEG-DASH content through an API REST[1]. Last but not least a MPEG-DASH client has been implement in order to make the reception and visualization of MPEG-DASH content in VOD (Video on Demand).

2. ARCHITECTURE AND FUNCTIONAL OVERVIEW

2.1. Publication and distribution

The publication and distribution component is a web application running in a Linux server using the Docker virtualization tool that is able to:

- List the content exported from the Premiere plugin and the content converted to MPEG-DASH
- Handle the MPEG-DASH transcoding of Premiere content
- Give user info about the transcode status of the Premiere content (Ready, Done, Processing, Exporting, Error)
- Publish MPEG-DASH content in order to view it through the multi-platform players.

The architecture of the web application server is divided into four modules. Each module is built as a Docker image and runs in a unique Docker container. The modules are: Nginx web server, web application, database and MPEG-DASH transcoder.

The web application is developed with Node JS[2] and AngularJS[3]. This module connects all the other modules. Processes like transcoding, publication, removal, etc. are executed via an API REST configured in the web application container managed by a NodeJS server. AngularJS is responsible of the web application's front-end. In conjunction with Bootstrap[4] it gives the web application a client side visual interface.

Input content, coming from Adobe Premiere, is shown in the "Convert" sub-menu. Output content, in MPEG-DASH format, is shown in the "Publish" sub-menu (See figure 1).

[1] Application program interface that uses HTTP requests to GET, PUT, POST and DELETE data.

[2] Open-source, cross-platform JavaScript runtime environment for developing a diverse variety of tools and applications. https://nodejs.org/en/

[3] JavaScript-based open-source front-end web application framework. https://angularjs.org

[4] Bootstrap is a free and open-source front-end web framework for designing websites and web applications. https://getbootstrap.com/

The database container uses MongoDB to save all the relevant information about the content and give additional information like conversion and publication status.

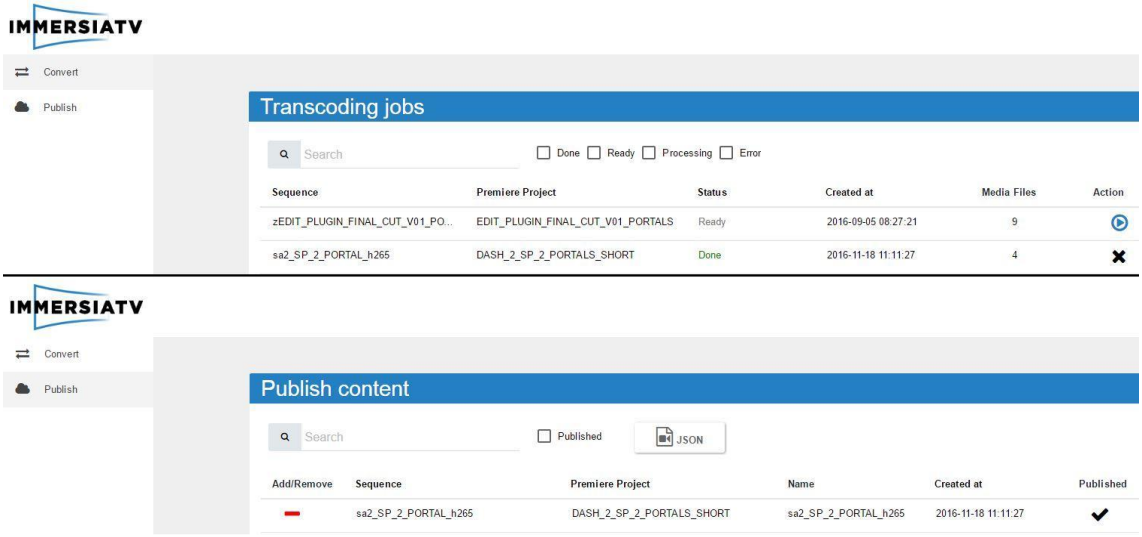

Figure 1: The distribution and publication web application view. Convert to MPEG-DASH section on top, publication content below.

MPEG-DASH transcoding is done by pressing the “Play” button under the Actions column. Doing this will activate the transcoder container via the API REST, and generate an output of different resolutions: HD (1080p) and UHD (2k, 4k) for the 360 and TV videos, and 420p for the portal videos. The final media set is (4k, 4k, 420p), (2k, 2k, 420p), (1080p, 1080p, 420p) with fragments that have a length of 3 seconds. The process is viewed through the status column. Once the conversion has ended the new DASH content will appear in the publish view. Once in the publish view, content can be published. This means it will be listed in the JSON file containing the list of content available.

The Nginx web server container connects the web application and the multi-platform players distributing the content. To provide a more robust infrastructure the content consumed by the players is delivered through the Nginx in a different port of the web application. This secures media distribution regardless of the web application’s state and therefore makes it more robust, particularly to the common DDoS attacks.

2.2. Client

The DASH client has been implemented using C language complemented with GObject and Glib libraries, which are the core languages in which the GStreamer[5] backend is implemented. The GStreamer Unity Bridge (GUB) implements the video reception and rendering pipeline, and it also includes the DASH client, the adaptive quality switching module and the video rendering module.

The GUB allows playing any media source in a Unity3D environment, and it constitutes one of the main components of the ImmersiaTV player (see, for further details, deliverable D3.6, which introduces the ImmersiaTV player). When the content is selected in the ImmersiaTV player, the GStreamer Client will obtain the DASH Manifest Uri.

The Manifest, called Media Presentation Description (MPD) is an XML document that describes different resolutions available for the player consumption. The implementation of the pipeline uses the element Playbin3, which is a GStreamer module that creates and combines dynamically the different elements needed to decode and reproduce content in different formats (DASH, mp4, or even RTMP or RTSP). It therefore provides a stand-alone and convenient abstraction for a generic audio and/or video player.

2.3. Future work

The main problem of content playout is selecting the content, because this depends of the device where the player is run. For example, the reproduction of 4K content gives a good experience on a TV, but is not appropriate for a Smartphone that has a screen resolution of 1080p.

Currently, the stream quality selection of the player follows strictly the DASH standard, and therefore only considers the estimation of the bandwidth (BW) available, at playout. The selection is based in the effective throughput of the BW available at the moment of download the MPEG-DASH content.

For this reason, the player needs to integrate an adaptive algorithm, to enable switching between the available representations of the media content depending on the device.

[5] https://gstreamer.freedesktop.org/

3. INSTALLATION GUIDE

3.1. Web application

MATERIALS

To install the ImmersiaTV distribution and reception service, you will need the following hardware components:

- A server (Windows, Mac OSX or Linux)
- Reception devices (Android TV Box Setup, Android mobile phones and/or a windows PC)

You will also need the following software:

- Docker
- ImmersiaTV web application installer YAML script
- ImmersiaTV players

PROCEDURE

Docker tools installation:

For Windows and Mac OSX, this comes in the package Docker Toolbox. To install it:

https://www.docker.com/products/docker-toolbox

Docker Toolbox is installed together with VirtualBox. In order for the server to work correctly we will have to open the next ports (8080, 8083) in:

- Settings > Network > Advanced > Port Forwarding (See figure 2)

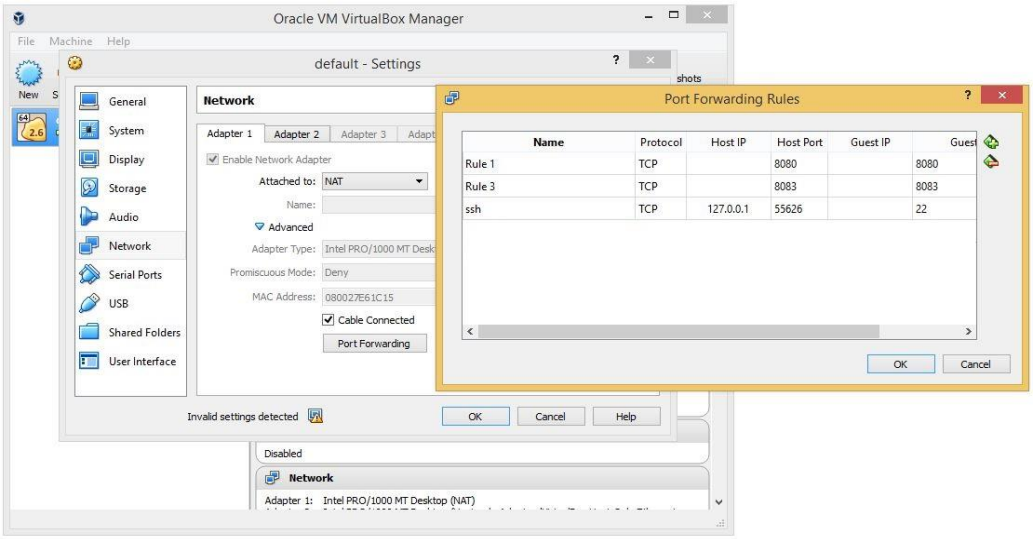

Figure 2: VirtualBox port opening menu interface

For Linux we will have to install Docker and Docker Compose following the instructions:

https://docs.docker.com/engine/installation/linux/ubuntulinux/
https://docs.docker.com/compose/install/

Ports don't have to be opened for this case.

In the host, where the web application is been deployed, create two folders in any known location:

- Folder for exported projects from Premiere (e.g. ~/User/input).
- Folder for the DASH content (e.g. ~/User/output)

Download the YAML script from the ImmersiaTV ftp server:

ftp://ftp.immersiatv.eu/releases/0.7/docker-compose.yml

Edit the **highlighted** parts with your personal settings:

```
nginx:
…
  volumes:
    - ~/User/input/:/data/dash
…
immersia:
…
  volumes:
    - ~/User/input/:/data/dash
    - ~/User/output/:/data/premiere
…
  environment:
    - HOST_IP=192.168.10.56
…
```

Figure 3: docker-compose.yml

Open a Terminal (Docker Terminal for Windows and Mac OSX), navigate to the path where the docker-compose.yml is located and insert:

```
docker-compose up
```

If something goes wrong press [CTRL + C] twice and insert the next command followed by the list of container names separate by spaces:

```
docker rm -f CONTAINER_NAME1 CONTAINER_NAME2 ...
docker-compose up
```

Navigate to http://[HOST_IP]:8080 and the ImmersiaTV content distribution web application has to be shown.

3.2. Client configuration

Follow the instructions for Client installation in Deliverable D3.6 Interaction and Display. In order to play the content published in section 2.1 Publication & Distribution:

1. Launch the app in your Smartphone.
2. When the app is running the player's main screen will be displayed (See figure 4):

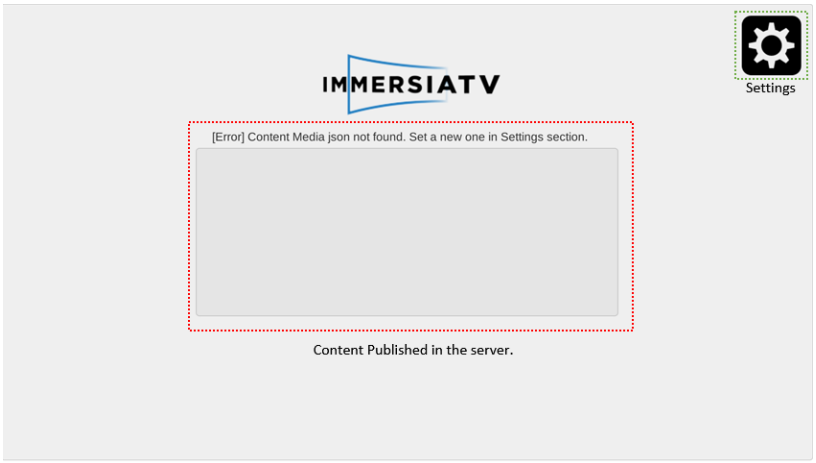

Figure 4: Player main screen

3. Tap the button settings, introduce the content media JSON's URI in the next format (See figure 5) and press connect:
   - http://[HOST_IP]:8080/dash/JSON_file

Figure 5: Player setting screen

4. If the connection to the JSON file is established, the published content will be listed (See figure 6):

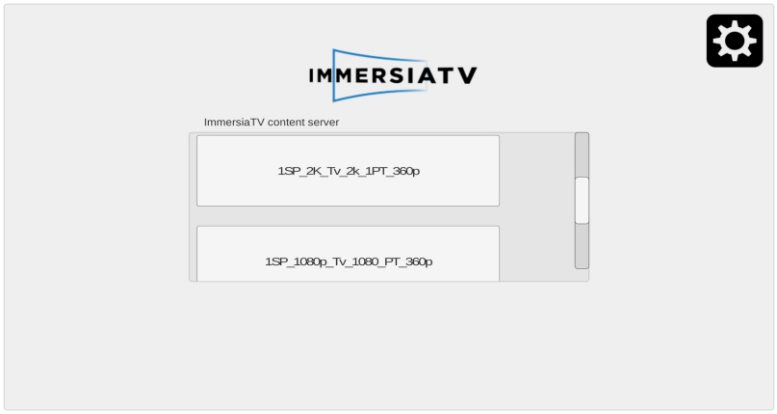

Figure 6: Player main screen showing the content published

5. Select one of the listed content and choose the viewing format between HMD, TV or Tablet (See figure 7).

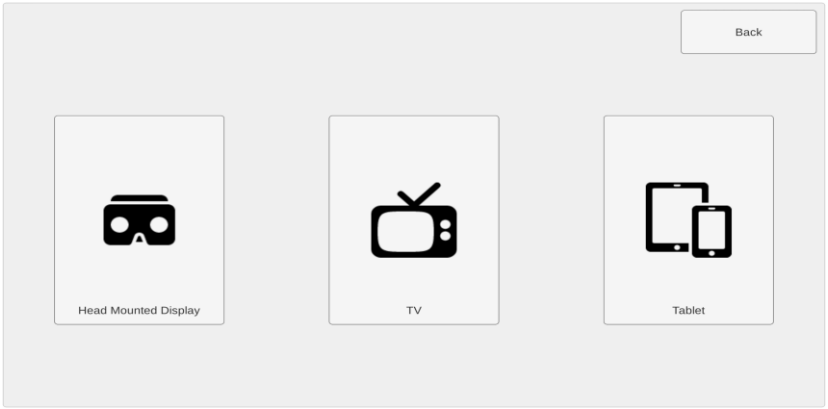

Figure 7: Player media content selection screen

4. REFERENCES

[1]. **International Organization of Standardization**. ISO/IEC 23009-1:2014. *Information technology - Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats.*