**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11/N17354**

**Gwangju, Korea – January 2018**

**Title: MPEG-I Architectures**

**Source: WG11**

**Status: Draft**

**Editor: Thomas Stockhammer, Imed Bouazizi**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Contents**

[1 Introduction 2](#_Toc504713887)

[2 Background 3](#_Toc504713888)

[3 Architectures 3](#_Toc504713891)

[4 Next Steps 9](#_Toc504713892)

# Introduction

This document gathers architectures for immersive media, predominantly focusing on 6 degrees of freedom applications. They are used to map use cases to the architectures and derive requirements to develop systems and formats for immersive 6DoF applications and services.

# Background

A key departure point from traditional 2D architectures is that visual rendering is governed by a graphics engine that composites the different media resources to create the presentation. Audio may undergo a similar procedure in the rendering.

In particular, the graphics engine will use tradition 2D content as texture for objects that are controlled by certain geometries. Physically-based rendering takes this approach to the extreme, where realistic light propagation, reflection/refraction patterns are mimicked with a high fidelity.

Most widely used graphics and game engines today rely on an OpenGL core (this may be different on some Operating Systems). They act as wrappers around OpenGL and offer more advanced functionality in a more abstract and simple to use way to the developers.

When rendering VR/AR or 6DoF content, the rendering engine usually sets up a scene first. The scene maybe read from a scene graph/scene description document or it may be inferred from the content (e.g. a scene with a single Sphere geometry for 360 video). The client may be given option to choose between a full 6DoF scene rendering, it may opt for a simplified rendering, or it may delegate part of the scene rendering to the network. In the latter case, the network will convert a 6DoF scene into a simplified 6DoF scene, a 3DoF+ or 3DoF scene, or even into a 2D video.

The media resources of a content may be of a wide range of formats and types. They can either be 2D or 3D, natural or synthetic, compressed or uncompressed, provided by the content provider or captured locally (e.g. in the case of AR).

A few key issues that are expected to be solved are:

* Definition of the spatial environment, i.e. the space in which the presentation is valid and can be consumed. Typically, for example what is referred to as windowed 6DoF, you have a limited amount of possible movements within the 3D space.
* presentation timeline management. The different resources may have an internal timeline for their presentation. The scene graph may have animations and other scripts that incur an internal media timeline. In addition, scene graphs should also be updateable in a 6DoF presentation, where updates are timed or event driven. Finally, the container format may also specify the media timeline for the presentation of the embedded media.
* Positioning and rendering of the media sources in the 6DoF scene appropriately. The individual media sources may for itself have descriptive metadata of their geometry or they may be described by the scene graphs. In any case such objects needs to be properly integrated in the 6DoF scene.
* Interacting with the scene based on sensor and/or user input. The rendered viewport can be depend on as simple aspects as the viewing position or may include complex sensor input or captured signals such as geolocation, gyrossope, temperature, camera out, eye tracking, etc.

Generally, the composition of timed media assets in a 6DoF VR or AR scene may be accomplished by one or more of the following means.

* By self-declarative media tracks that directly map themselves into a common reference system as for example done by OMAF
* By an MPEG defined scene description that is expected to be carried along with the timed media assets.
* By an external scene description that is included as a track in order to compose the contained media assets properly.

# Architectures

## Introduction

This clause documents different architectures that may be of relevance for 6DoF Immersive Media Services. The key focus is a device architecture, but also network architectures and interfaces are considered. These architectures are collected in order for use case proponents to make use of such an architecture and define the usage and instantations for relevant use cases. Once the use cases are mapped, the relevant interfaces and APIs can be extracted and normative standardization requirements can be defined.

## Architecture Components

### Introduction

In order to structure the work, architectural components are defined. We separate device architectures as well as network architectures.

### Network Components

Control and Management Function: A function that establishes the communication between an MPEG-capable client and an MPEG-capable network element, or multiple of those. It is expected that such a management function is out of scope for MPEG standards unless a very clear requirement would be developed.

MPEG Content Origin: A server (may be centralized or distributed) that hosts MPEG formatted content and/or can be accessed with MPEG-defined protocols.

MPEG Media Aware/Processing Network Function: A function that communicates with either MPEG Content Origin or the MPEG-capable client or both in order to support the client in experiencing the immersive scene.

MPEG-capable Client: A function that provides all means to consume an immersive MPEG content.

### Device Components

* Lower Layer (5G, WiFi, IP)
* Media Access Client (DASH Client, MMT, File System)
  + Extracts elementary streams and metadata and makes it available to decoders and Applications
* Uplink Client
  + Uplink Media
    - Media encoder
    - Content Delivery Protocol
  + Metadata
* Decryption
* Media Decoders
  + Audio
  + Video
* Application/Presentation Engine
  + SceneGraph
  + Management
  + SceneDescription
  + Scripts
  + Timed Track for Application/Presentation Engine
* Rendering
  + Audio
  + Visual
* XR Functions
  + XR Input: Buttons, Sticks, Triggers, Tracking
  + XR Display
  + XR Compositor
  + Sensor Data
* Capture:
  + Microphones
  + Camera
  + Others

## Traditional Architecture and Phase 1 Architecture: Passive consumption centric

The architecture of the first phase is based on OMAF as shown in Figure 1 including the key specification area for OMAF. The key specifications are on formats such that a device and applications can be build that permit

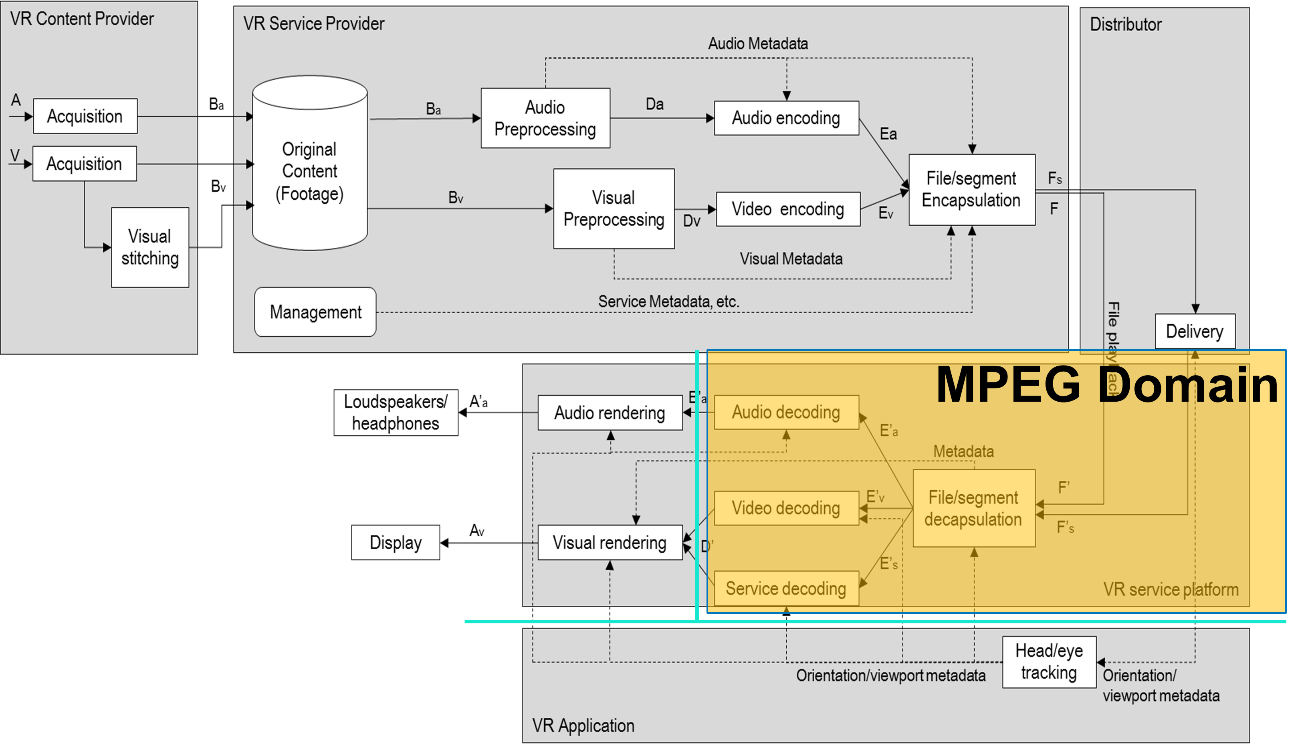


Figure 1 OMAF Architecture and the key MPEG domain specifications

In order to structure the work, we should look at typical properties in the specification and the assumption around this, and based on this we should build further on extending the architecture into different directions addressing the different use cases. Starting from a traditional architecture and system model in MPEG as done for MPEG-2 TS, DASH, ISO BMFF or probably also MMT, the system standard provides primarily:

1. Timing and synchronization of different media streams
2. Multiplexing or at least combination of the media streams
3. System functions for consuming the content such as random access
4. Description on the required capabilities to consume the stream
5. Description on role/property of the streams (language, role, etc.)
6. Ability to deliver the content on different network scenarios: Adaptation, Switching, error resilience
7. Other system functions such as encryption

The ability for a user/interface to “interact” with traditional content is limited to random access, seeking and component selection. This selection/interaction impacts not only the media playback, but also media decoding and delivery and hence formats and communication aspects are involved. Typically, the consumption is passive.

Such stream definition does not prevent to that media is included in richer interaction environments (for example an interactive framework such as HTML-5 and browser consumption) as shown in Figure 2, but this not part of the MPEG architecture and system part.

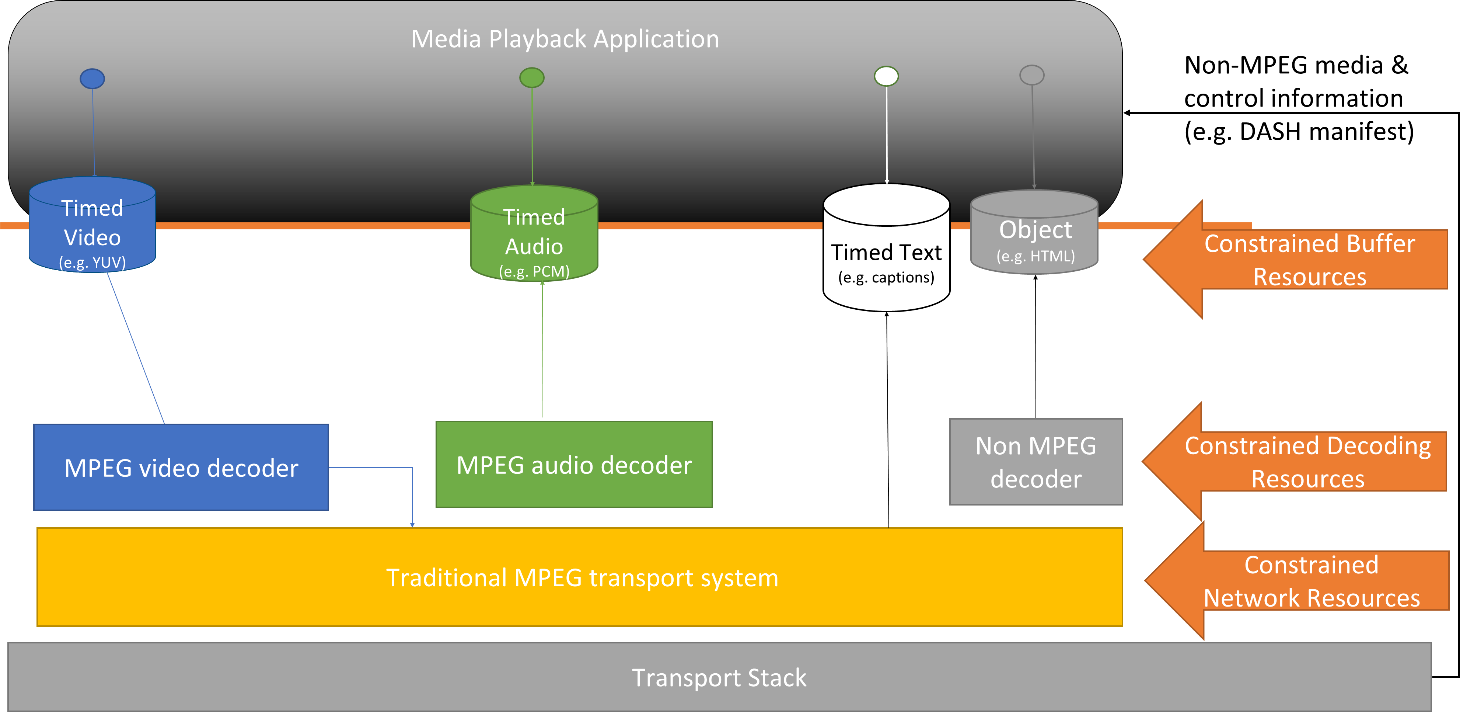


Figure 2 Media Playback using DASH in HTML-5 with media elements

In MPEG-I phase 1, the architecture was extended that the application interacts with the media stack by providing also the viewport. This interaction results in additional dynamics and optimizations. The viewport is multidimensional and includes sensor data (viewing information) and device capabilities (viewport size) as shown in Figure 3. Other than this, the architecture is relatively unchanged, and a consumption based model is considered, i.e. the amount of uplink information is low and does not involve media.

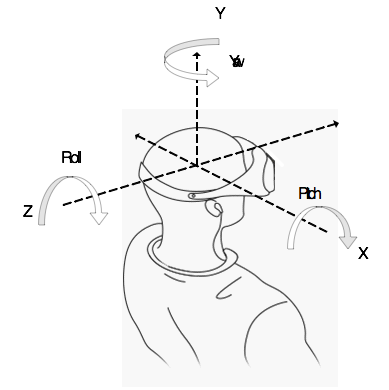
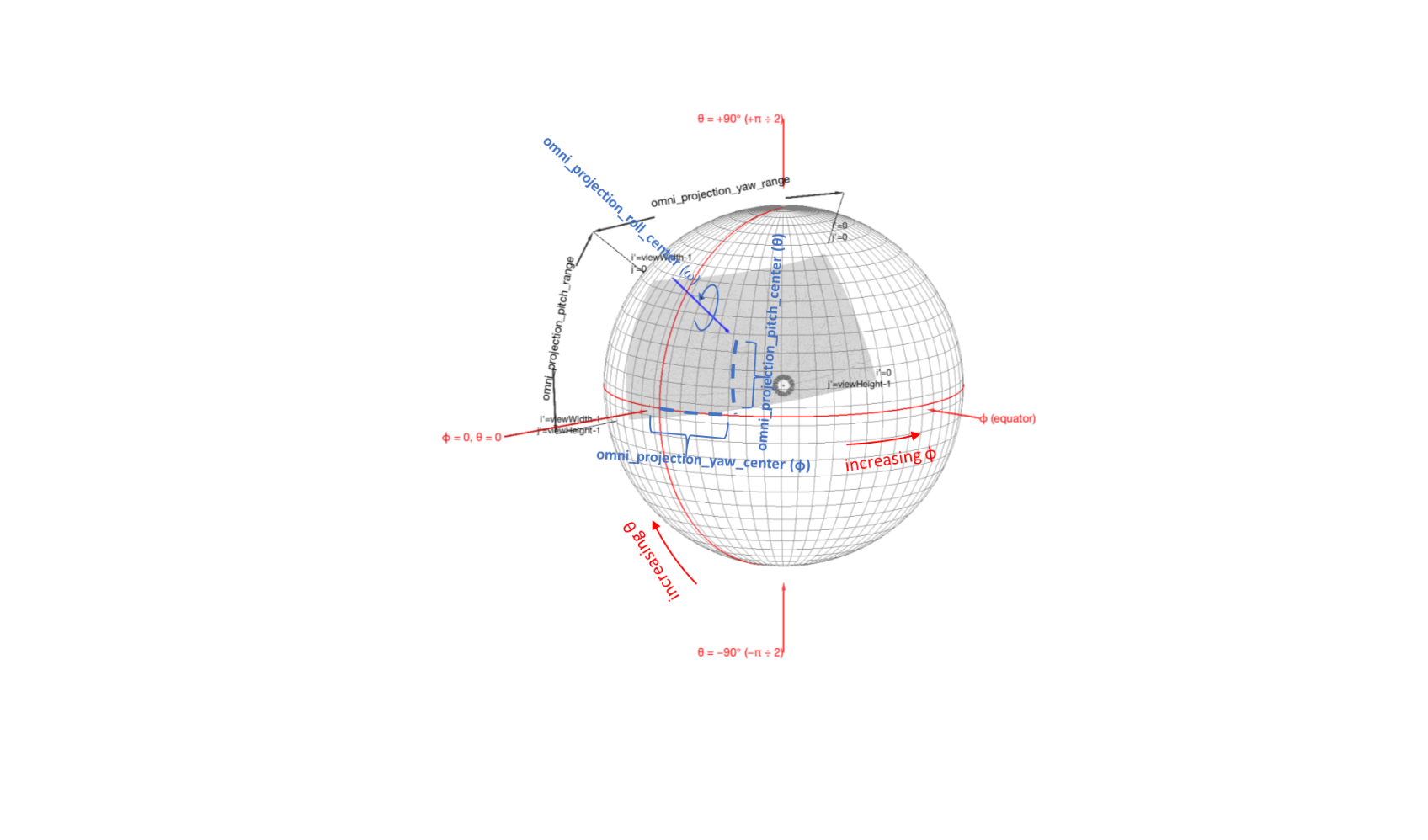
 

Figure 3 Adding sensor data to consumption

This architecture may be maintained, but for example for 3DoF+ additional sensor data may be added, for example a certain amount of depth or translational movement.

## Phase 2 architecture: Interaction and Rendering Centric

This architecture foresees input through different modalities at the client to account for applications such as 3DoF/6DoF and even AR/MR. Input from local cameras and microphones is usually required for AR/MR applications.

Rendering is performed by a Graphics Rendering Engine for visual content and a 2D/3D Audio Rendering Engine for audio content. The Graphics Rendering Engine is usually based on some graphics libraries such as OpenGL or WebGL or even some higher-level engines such as Unity. Decoded media resources are composited together by the rendering engine to produce the Presentation.

The architecture supports MPEG and non-MPEG media resources. These can be timed and non-timed. The container parser extracts information about resources and media timeline as well as any embedded or referenced media resources and make them available at the presentation engine.

The Architecture supports consuming the content in different forms, e.g. a simplified 2D version may be rendered in simple clients, a limited 3DoF, 3DoF+, or 6DoF version may also be consumed by the client. This pre-rendering/simplification may be done locally or in the network in a pre-rendering (baking) step. This is necessary for clients that are limited in their processing capabilities or network resources. Alternatively, the full-fledged 6DoF presentation may be consumed by clients capable of consuming it and having the required network resources and processing power.

The rendering engine may compose 3D content out from 2D-content. An example is point clouds that are encoded using MPEG-encoded Point Cloud Compression.

To describe the scene of a presentation, a scene graph may be used. The scene graph may be provided in alternative formats to offer the renderer the choice of picking a supported scene graph format. Alternatively, a basic rendering operation may be described in the container format to support simple clients that do not support any of the included scene graph formats. Other scene description files such as scripts or shaders may also be included in the container.

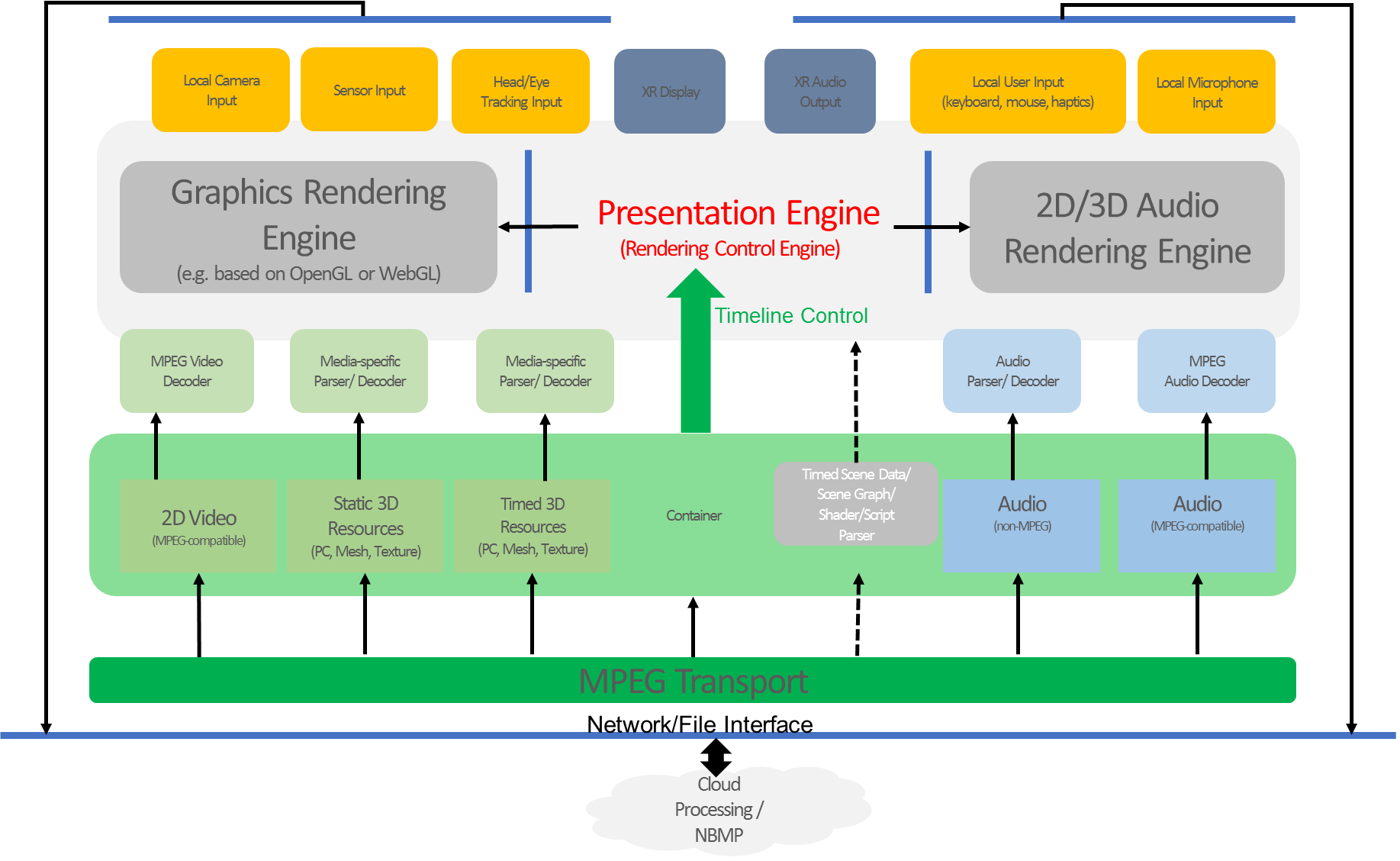


Figure 4 Consider draft Phase 2 Reference architecture

Some other aspects and alternatives have been considered and are maintained for now.

At MPEG#120 the extension and combination of MPEG media assets with interactive presentations was discussed extensively. An example is the usage of Scene Graph to describe the combination of media assets, in a similar way as for example media elements are placed into an HTML-5 presentation as shown in Figure 4. The interaction is handled by the application and the metadata to make use of the MPEG media is part of the interactive framework.

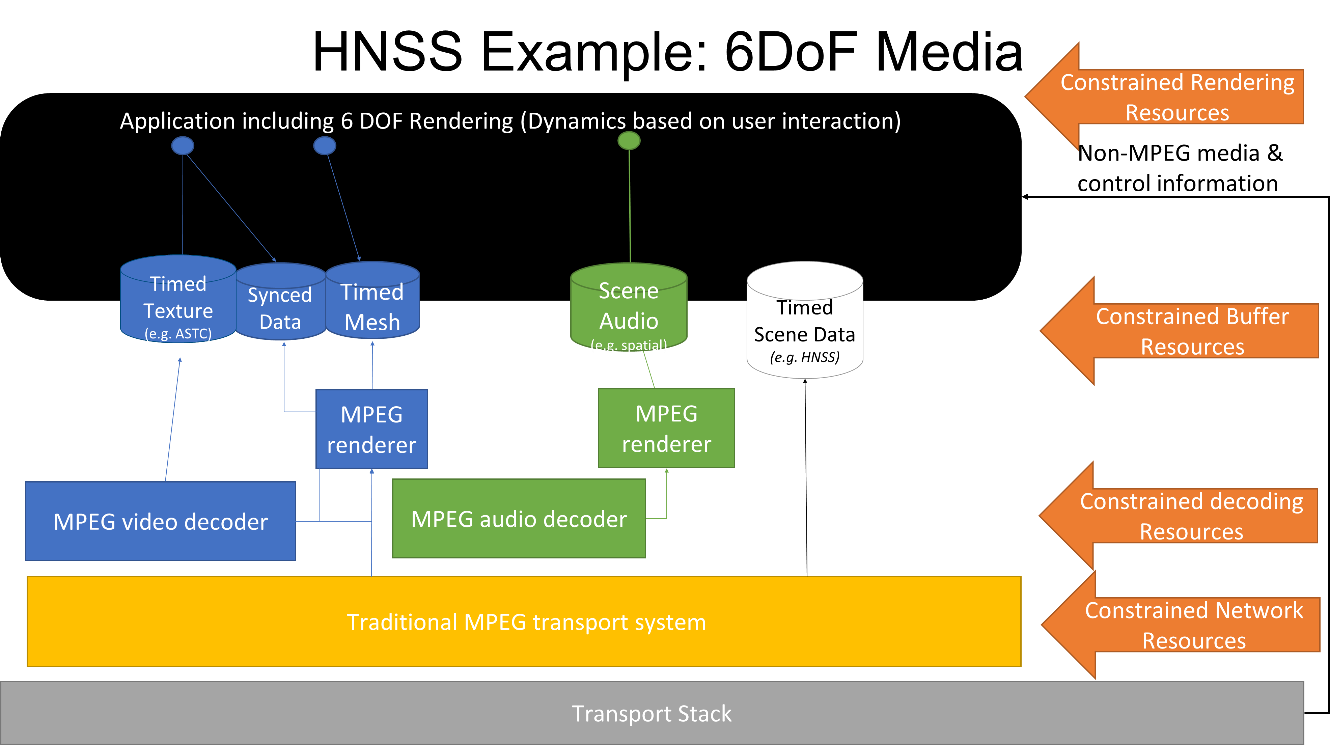


Figure 4 Scene Graph for potential 6DOF of freedom presentation.

As a second aspect, the issue of network rendering was included in the discussion as shown in Figure 5.

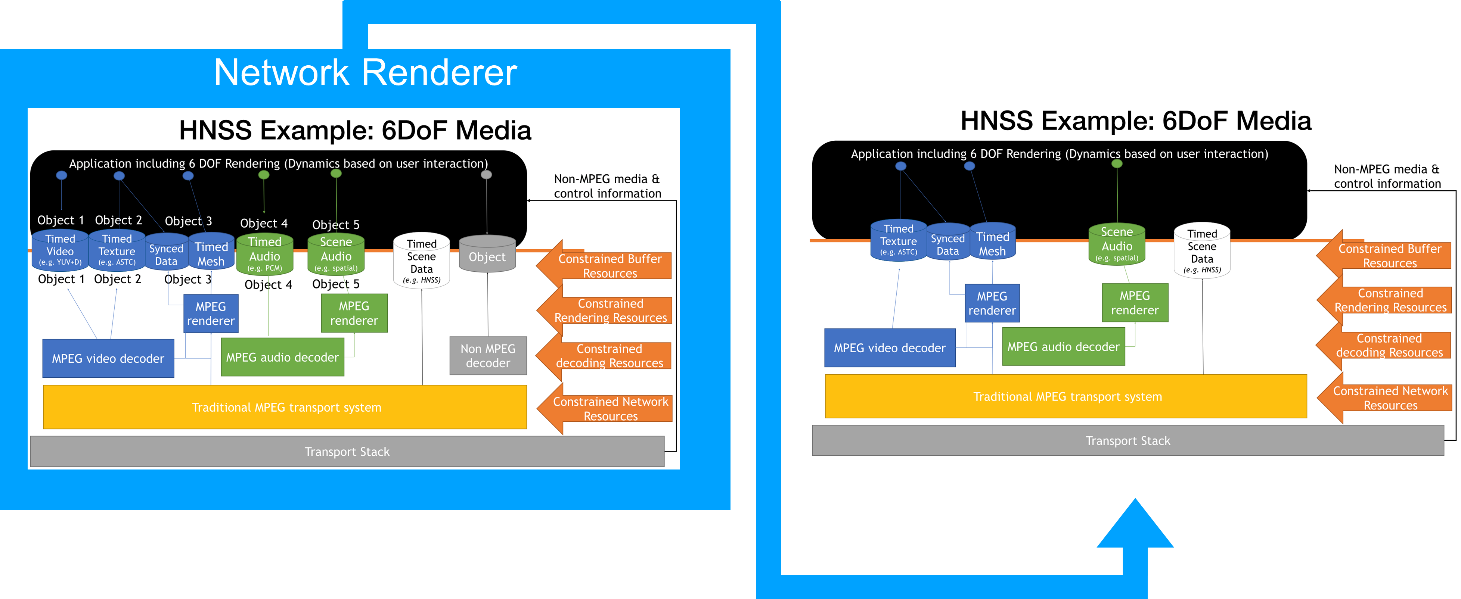


Figure 5 Including network Rendering for 6DoF

As a summary the architecture from the last MPEG meeting is extended to add some AR aspects in Figure 6

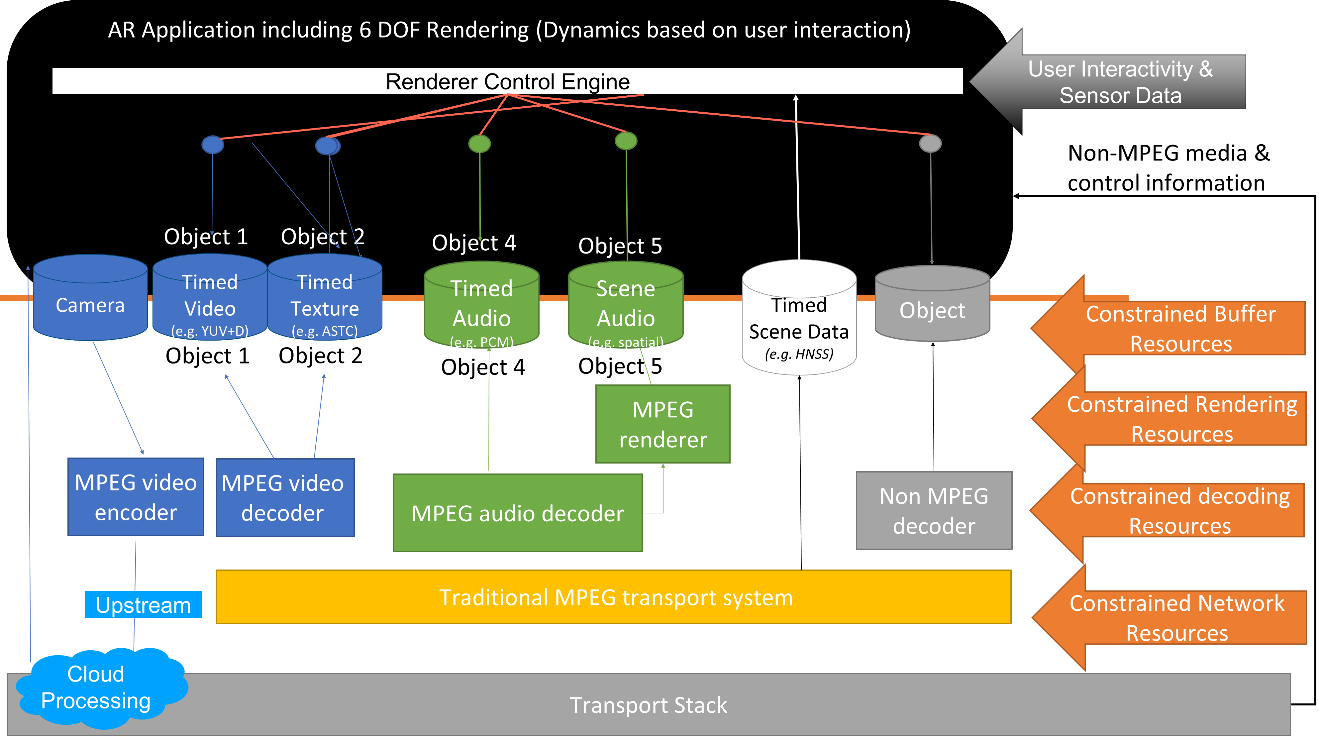


Figure 12 AR extensions for the protocol stack

# Next Steps

Based on the initial discussion in clause 2 and 3, it is expected that

* the use cases documented for phase 2 are mapped to the documented architectures using instantations and call flows
* If the architecture is not sufficient, the architectures are extended and modified
* A modified and harmonized set of architectures is developed in order to address the different use cases
* The individual functions are identified and well defined
* The relevant interfaces are named and options for fulfillments are provided
* The potential standards-relevant interfaces are identified and gaps are analyzed.