Moving Picture, Audio and Data Coding by Artificial Intelligence
www.mpai.community

N1536		2023/12/20
Source	Requirements (OSD)	
Title	Technical Specification – Object and Scene Description (MPAI-OSD) WD for Community Comments	
Target	MPAI-39	

This document is a Working Draft published with a request for Comments. Anybody may submit comments to the MPAI Secretariat at secretariat@mpai.community. Comments will be considered in the final draft of MPAI-OSD V1 if received by 2024/01/25T23:59 UTC.

Moving Picture, Audio and Data Coding
by Artificial Intelligence
www.mpai.community

MPAI Technical Specification

Object and Scene Description

WD for Community Comments

WARNING

Use of the technologies described in this Technical Report may infringe patents, copyrights, or intellectual property rights of MPAI Members or non-members.

MPAI and its Members accept no responsibility whatsoever for damages or liability, direct or consequential, which may result from the use of this Technical Report.

Readers are invited to review Annex 3 - Notices and Disclaimers.

Technical Specification
Object and Scene Description
(under development)

1 Introduction

Technical Specification: Object and Scene Description (MPAI-OSD) – in the following also called MPAI-OSD – has been developed by MPAI – Moving Picture, Audio, and Data Coding by Artificial Intelligence, the international, unaffiliated, non-profit organisation developing standards for Artificial Intelligence (AI)-based data coding with clear Intellectual Property Rights licensing frameworks in compliance with the rigorous MPAI Process [9,10] in pursuit of the following policies:

1. Be friendly to the AI context but, to the extent possible, agnostic to the technology – AI or Data Processing – used in an implementation.
2. Be attractive to different industries, end users, and regulators.
3. Address three levels of standardisation all exposing standard interfaces with an aggregation level decided by the implementer:
   3.1. data types.
   3.2. Components called AI Modules (AIM).
   3.3. Configurations of AIMs called AI Workflows (AIW).
4. Specify the data exchanged by AIMs with as clear a semantic as possible.

As manager of the MPAI Ecosystem specified by Governance of MPAI Ecosystem (MPAI-GME) [1] and ensures that a user can:

1. Operate a reference implementation of the Technical Specification, by providing a Reference Software Specification with annexed software.
2. Test the conformance of an implementation with the Technical Specification, by providing the Conformance Testing Specification.
3. Assess the performance of an implementation of a Technical Specification, by providing the Performance Assessment Specification.
4. Get conforming implementations possibly with a performance assessment report from a trusted source through the MPAI Store.

Technical Specification: AI Framework (MPAI-AIF) [2] specifies a standard AI Framework (AIF) that enables dynamic configuration, initialisation, and control of AIWs depicted in Figure 1.

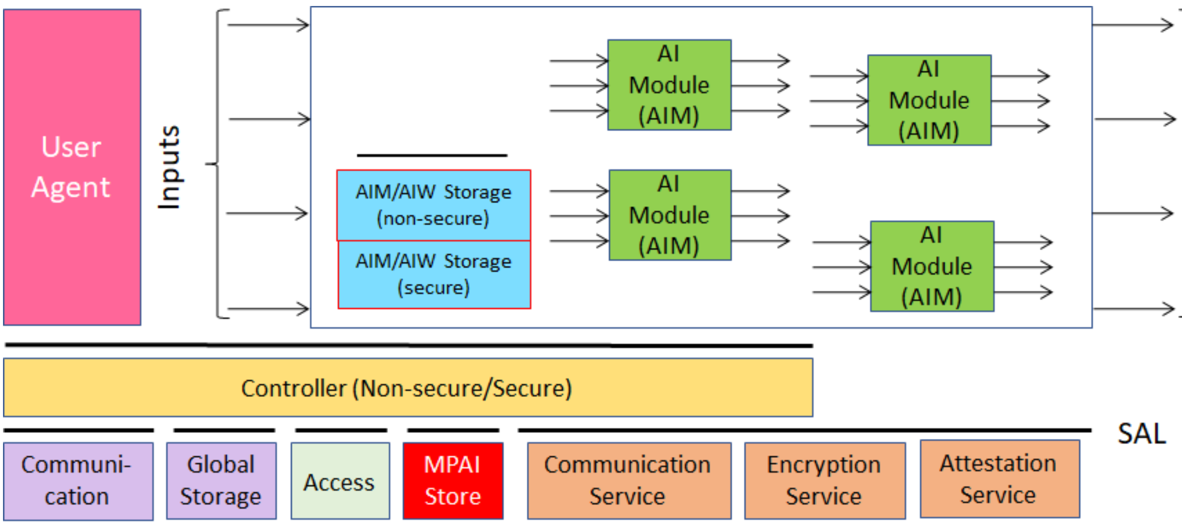

Figure 1 - The AI Framework (MPAI-AIF) V2 Reference Model

MPAI-AIF enabling the secure execution of AI Workflows (AIW) that can be constituted by AI Modules (AIM). Thus, users can have machines whose internal operation they understand to some degree, rather than machines that are just “black boxes” resulting from unknown training with unknown data and component developers can provide components with standard interfaces that can have improved performance compared to other implementations.

An AIW and its AIMs may have 3 interoperability levels:
Level 1 – Implementer-specific and satisfying the MPAI-AIF Standard.
Level 2 – Specified by an MPAI Application Standard.
Level 3 – Specified by an MPAI Application Standard and certified by a Performance Assessor.
Users are free to adopt any of the three levels.

AIM can execute data processing or Artificial Intelligence algorithms and can be implemented in hardware, software, or hybrid hardware/software. AI Module can be Composite if they include connected AI Modules.

The MPAI-MMC V2 Technical Specification can be implemented in one of the following modalities:
1. As a specific AIW implementing a Use Case, as specified in this document.
2. As a specific AIM, as specified in this document.
3. As a specific data type, as specified in this document.

However, MPAI does not mandate the choice of modality, which remains the sole decision of the implementer.

In many MPAI Technical Specifications there are data types that refer to Objects and Scenes that can be uni- and multimodal and possibly refer to locations that may be in a physical or virtual space.

MPAI values the consistent use of data types across its Technical Specifications. Therefore, MPAI-OSD has been developed to be the reference point for the consistent use of data types across MPAI standards. When consistency is not possible because different usages are consolidated, such usages are clearly identified, and individual specific usages recorded.

Currently, there are no MPAI-OSD specific Use Cases. Therefore, only the Scope, Reference Models, and I/O Data of relevant Use Cases from other Technical Specifications are reported here. The full Use Case specification can be found in the Technical Specifications owning the Use Cases. All Use Cases are assumed to be implemented according to the MPAI-AIF.

MPAI-OSD will be accompanied by the Reference Software, Conformance Testing, and Performance Assessment Specifications. Conformance Testing specifies methods enabling users to ascertain whether a data type generated by an AIM, an AIM, or an AIW conform with this Technical Specification.

2 Scope

Technical Specification: Object and Scenes Description (MPAI-OSD) specifies Data Formats to enable description and localisation of uni- and multi-modal Objects and Scenes in a Virtual Space for uniform use across MPAI Technical Specifications.

MPAI-OSD has been developed by the Context-based Audio Enhancement (MPAI-CAE), Multimodal Conversation (MPAI-MMC), and by the Portable Avatar Format (MPAI-PAF) Development Committees, and by the Connected Autonomous Vehicle (CAV) group of the Requirements Standing Committee.

3 Definitions

Terms beginning with a capital letter have the meaning defined in Table 1. Terms beginning with a small letter have the meaning commonly defined for the context in which they are used. For instance, Table 1 defines *Object* and *Scene* but does not define *object* and *scene*.

A dash "-" preceding a Term in Table 1 indicates the following readings according to the font:

1. Normal font: the Term in the table without a dash and preceding the one with a dash should be read before that Term. For example, "Avatar" and "- Model" will yield "Avatar Model."
2. *Italic* font: the Term in the table without a dash and preceding the one with a dash should be read after that Term. For example, "Avatar" and "- Portable" will yield "Portable Avatar."

Table 1 - General MPAI-HMC terms

Audio	Digital representation of an analogue audio signal sampled at a frequency between 8-192 kHz with a number of bits/sample between 8 and 32, and non-linear and linear quantisation. Data with characteristics of Audio may be synthetically produced.
Avatar	An Object rendered to represent a Human of a Machine in a virtual space.
- Model	An inanimate Avatar exposing animation interfaces.
- *Portable*	A Data Type including Avatar ID, Time, Audio-VisualScene Descriptors, Spatial Attitude, Avatar Model, Body Descriptors, Face Descriptors, Language Preference, Speech Coding, Speech Data, Text, and Personal Status [5].
Centre Point	The point of an Object selected to have coordinates (0,0,0).
Data	Information in digital form.
- Format	The standard digital representation of Data.
- Type	An instance of Data with a specific Data Format.
Descriptor	The Digital Representation of a feature of an Object.
- *Body*	A Data Type including the digital representation of the features of the body of a real or digital human.
- *Face*	A Data Type including the digital representation of a feature of the face of a real or digital human.
Digital Representation	Data corresponding to and representing a physical entity.
Environment	A Virtual Space that may be null or may include an Audio-Visual Scene.
Human	A human being in a real space.
- *Digital*	A Digitised or a Virtual Human.
- *Digitised*	An Object that has the appearance of a specific human when rendered.
- *Virtual*	An Object created by a computer that has a human appearance when rendered but is not a Digitised Human.
Identifier	The label uniquely associated with a human or an Object.
Instance	An element of a set of entities – Objects, Digital Humans etc. – belonging to some levels in a hierarchical classification (taxonomy).
- *Audio*	The instance of an Audio Object.
- *Visual*	The instance of a Visual Object.
Object	A data structure that can be rendered to cause an Experience.
- *Audio*	An Object described by Audio Descriptors.
- *Audio-Visual*	An Object described by Audio-Visual Descriptors.
- *Body*	A digital representation of the body of a Human or a Machine.

- *Descriptor*	The digital representation of the feature of an Object.
- *Digital*	A Digitised or a Virtual Object.
- *Digitised*	The digital representation of a real object.
- *Face*	The digital representation of the face of a Human or a Machine.
- *Speech*	An Object described by Speech Descriptors.
- *Text*	A string of Text.
- *Virtual*	An Object not representing an object in the real environment.
- *Visual*	An Object described by Visual Descriptors.
Orientation	The 3 Euler angles of an Object in a Virtual Space.
Position	The coordinates of a representative point for an object in a Virtual Space with respect to a set of coordinate axes.
Rendering	The process of instantiating a Virtual Space as a human-perceptible entity.
Scene	A composition of Objects located according to a Scene Geometry.
- *Audio*	A Scene composed of Audio Objects.
- *Digital*	A digitised scene or a Virtual Scene
- *Audio-Visual*	A Scene composed of Audio Objects, Visual Objects and co-located Audio-Visual Objects.
- *Visual*	A Scene composed of Visual Objects.
Scene Descriptors	The digital representation of a feature of a scene.
- *Audio*	A Data Type including the digital representation of the audio features of a digital scene.
- *Audio-Visual*	A Data Type combining the Audio or Visual Scene Descriptors.
- *Visual*	A Data Type including the digital representation of the visual features of a digital scene.
Scene Geometry	The digital representation of the Object arrangement of a Scene.
- *Audio*	A Data Type describing the Spatial arrangement of the Visual Objects of a Scene.
- *Audio-Visual*	A Data Type describing the Spatial arrangement of the Audio, Visual, and Audio-Visual Objects of a Scene.
- *Visual*	A Data Type describing the Spatial arrangement of the Visual Objects of a Scene.
Attitude	
- *Spatial*	Position and Orientation and their velocities and accelerations of a Human and Physical Object in a Real or Virtual Environment.
Virtual Space	A space generated and maintained by a computing platform that can be rendered.
Speech	Digital representation of analogue speech sampled at a frequency between 8 kHz and 96 kHz with a number of bits/sample of 8, 16 or 24, and non-linear and linear quantisation or compressed. Data with characteristics of Speech may be synthetically produced.

4 References

4.1 Normative Reference

1. MPAI; Technical Specification: Governance of the MPAI ecosystem (MPAI-GME), V1.1; https://mpai.community/standards/mpai-gme/
2. MPAI; Technical Specification: AI Framework (MPAI-AIF) V2; https://mpai.community/standards/mpai-aif/

3. MPAI; Technical Specification: Context-based Audio Enhancement (MPAI-CAE) V2.1; https://mpai.community/standards/mpai-cae/
4. Technical Specification: Connected Autonomous Vehicles (MPAI-CAV) – Architecture V1; https://mpai.community/standards/mpai-cav/
5. MPAI; Technical Specification: Multimodal Conversation (MPAI-MMC) V2; https://mpai.community/standards/mpai-mmc/
6. Technical Specification: MPAI Metaverse Model (MPAI-MMM) – Architecture V1; https://mpai.community/standards/mpai-mmm/
7. MPAI; Technical Specification: Portable Avatar Format (MPAI-PAF) V1; https://mpai.community/standards/mpai-paf/
8. Khronos; Graphics Language Transmission Format (glTF); October 2021; https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html

4.2 Informative References

9. MPAI; The MPAI Statutes; N421; https://mpai.community/statutes/
10. MPAI; Patent Policy; https://mpai.community/about/the-mpai-patent-policy/
11. MPAI; Framework Licence: Object and Scene Description; https://mpai.community/wp-content/uploads/2023/08/N1361-Framework-Licence-Object-and-Scene-Description-MPAI-OSD.pdf

5 Use Cases

This Technical Specification refers to Use Cases of other MPAI Technical Specifications that are relevant to this Technical Specification. Only the Scope, Reference Model, and I/O Data of the Use Cases are reported here. In case of discrepancy between a Use Case reported here and the one in the Technical Specification owning it, the latter prevails.

5.1 Conversation About a Scene (CAS)

The full specification of this Use Case is provided in [5].

5.1.1 Scope of Conversation About a Scene

This Use Case addresses the case of a human holding a conversation with a mMchine:

1. The Machine sees and hears an Environment containing a speaking human and some scattered objects.
2. The Machine recognises the human's Speech and obtains the human's Personal Status by capturing Speech, Face, and Gesture.
3. The human converses with the Machine indicating the object in the Environment s/he wishes to talk to or ask questions about it using Speech, Face, and Gesture.
4. The Machine understands which object the human is referring to and generates an avatar that:

4.1. Utters Speech conveying a synthetic Personal Status that is relevant to the human's Personal Status as shown by his/her Speech, Face, and Gesture, and

4.2. Displays a face conveying a Personal Status that is relevant to the human's Personal Status and to the response the Machine intends to make.

5. The Machine displays the Scene Presentation corresponding to how it perceives the Environment from a human-selected Point of View. The objects in the scene are labelled with the Machine's understanding of their semantics so that the human can understand how the Machine sees the Environment.

5.1.2 Reference Architecture of Conversation About a Scene

Figure 2 gives the Conversation About a Scene Reference Model including the input/output data, the AIMs, and the data exchanged between and among the AIMs.

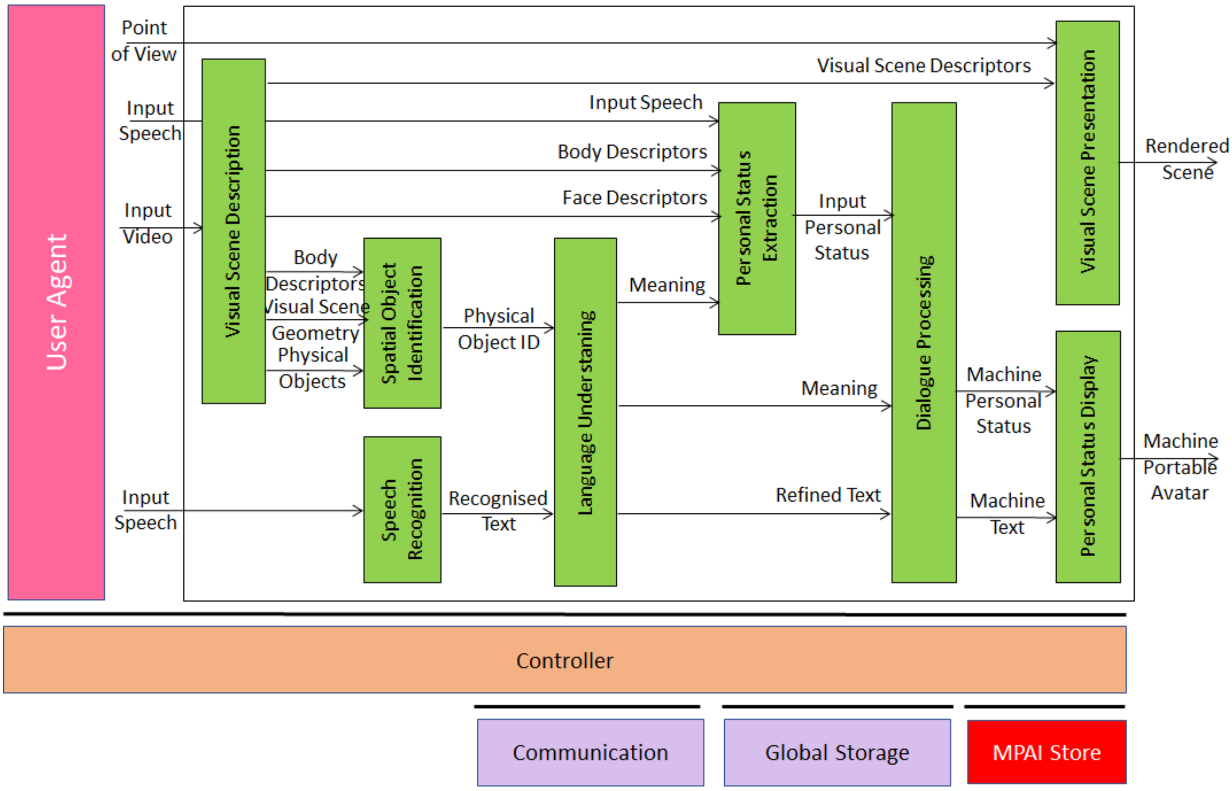

Figure 2 – Reference Model of Conversation About a Scene

The Machine operates according to the following workflow:

1. Visual Scene Description produces Body Descriptors, Visual Scene Geometry and Physical Objects from Input Video.
2. Speech Recognition produces Recognised Text from Input Speech.
3. Spatial Object Identification produces Physical Object Instance ID from Physical Objects, Body Descriptors, and Visual Scene Geometry.
4. Language Understanding produces Meaning and Refined Text from Recognised Text and Physical Object ID.
5. Personal Status Extraction produces Input Personal Status from Meaning, Input Speech, Face Descriptors, and Body Descriptors.
6. Dialogue Processing produces Machine Text and Machine Personal Status from Input Personal Status, Meaning, and Refined Text.
7. Personal Status Display produces Machine Portable Avatar from Machine Text, and Machine Personal Status.
8. Scene Presentation uses the Visual Scene Descriptors to produce the Rendered Scene as seen from the user-selected Point of View. The rendering is constantly updated as the machine improves its understanding of the scene and its objects.

5.1.3 I/O Data of Conversation About a Scene

Table 2 gives the input/output data of Conversation About a Scene.

Table 2 – I/O data of Conversation About a Scene

Input data	From	Comment
Input Video	Camera	Points to human and scene.
Input Speech	Microphone	Speech of human.
Point of View	Human	The point of view of the scene displayed by Scene Presentation.
Output data	**To**	**Comments**
Rendered Scene	Human	Rendering of the scene containing labelled objects as perceived by Machine and seen from the Point of View.
Machine Portable Avatar	Human	Machine's avatar.

5.2 Human-Connected Autonomous Vehicle (CAV) Interaction (HCI)

Reference [4] provides the full specification of this Use Case.

5.3 Functions of Human-CAV Interaction

The Human-CAV Interaction (HCI) Subsystem performs the following high-level functions:

1. Authenticates humans e.g., for the purpose of letting them into the CAV.
2. Interprets and executes commands provided by humans, possibly after a dialogue, e.g., to go to a Waypoint, issue commands such as turn off air conditioning, open window, call a person, search for information, etc.
3. Displays Full Environment Representation to passengers via a viewer and allows passengers to navigate and control the viewing.
4. Interprets conversation utterances with the support of the extracted Personal Statuses of the humans, e.g., on the fastest way to reach a Waypoint because of an emergency, or during a casual conversation.
5. Displays itself as a Body and Face with a mouth uttering Speech showing a Personal Status comparable to the Personal Status that a human counterpart (e.g., driver, tour guide, interpreter) would display in similar circumstances.

The HCI operation is highly influenced by the notion of *Personal Status*, the set of internal characteristics of conversation humans and machines.

5.4 Reference Architecture of Human-CAV Interaction

Figure 3 gives the Human-CAV Interaction (HCI) Reference Model supporting the case of a group of humans approaching the CAV from outside the CAV and sitting inside the CAV.

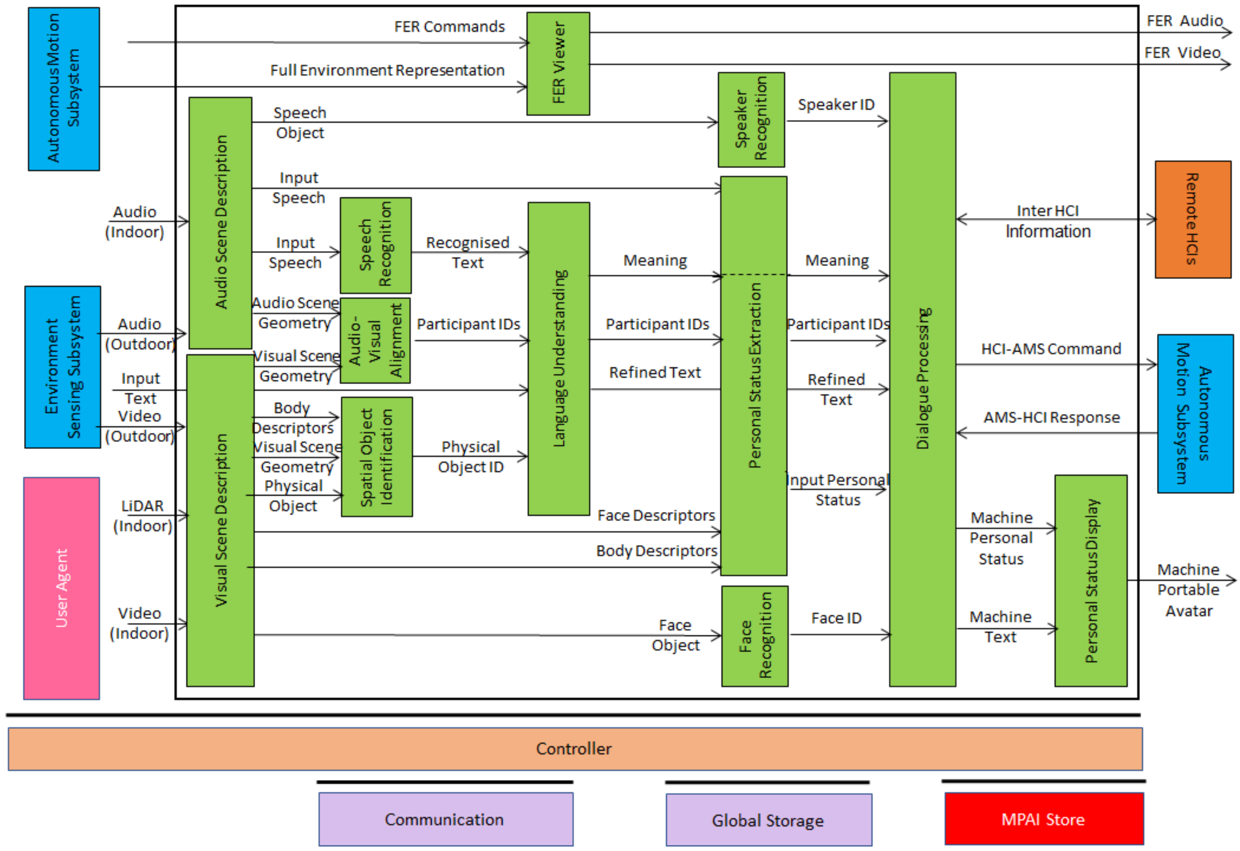

Figure 3 – Human-CAV Interaction Reference Model

The HCI operation is considered in two outdoor and indoor human-CAV interaction scenarios:

1. Audio Scene Description AIM creates the Audio Scene Description in the form of 1) Audio (Speech) Objects corresponding to each speaking human in the Environment (close to the CAV) and 2) Audio Scene Geometry.
2. Visual Scene Description creates the Visual Scene Descriptors in the form of Descriptors of 1) Faces and the Bodies corresponding to each human in the Environment (close to the CAV) and 2) Visual Scene Geometry.
3. Speech Recognition recognises the speech of each human.
4. Spatial Object Alignment Identifies Audio, Visual, and Audio-Visual Objects, from Audio and Visual Scene Geometries.
5. Spatial Object Identification produces Object ID from Physical Objects, Body Descriptors, and Visual Scene Geometry.
6. The Full Environment Representation (FER) Viewer renders the FER in response to FER navigation Commands.
7. Language Understanding produces the Refined Text and extracts the Meaning.
8. The Speaker Recognition and Face Recognition AIMs authenticate the humans the HCI is interacting with. The processing of these two AIMs may be carried out remotely.
9. The Personal Status Extraction AIM extracts the Input Personal Status from Meaning, Speech, Face Descriptors, and Body Descriptors.
10. The Dialogue Processing AIM:

10.1. Validates the human Identities.

10.2. Produces Machine Text and Machine Personal Status.

11. The Personal Status Display produces the ready-to-render Machine Portable Avatar [7] conveying Machine Speech and Machine Personal Status.

11.1. Issues commands to the Autonomous Motion Subsystem.

11.2. Receives and processes responses from the Autonomous Motion Subsystem.
11.3. Communicates with Remote HCIs.

5.5 I/O Data of Human-CAV Interaction

Table 3 gives the input/output data of the Human-CAV Interaction Subsystem.

Table 3 – I/O data of Human-CAV Interaction

Input data	From	Comment
Full Environment Representation	Autonomous Motion Subsystem	Rendered by Full Environment Representation Viewers
Full Environment Representation Commands	Cabin Passengers	To control rendering of Full Environment Representation
Audio (Outdoor))	Environment Sensing Subsystem	User authentication User command User conversation
Audio (Indoor)	Cabin Passengers	User's social life Commands/interaction with HCI
Video (Outdoor)	Environment Sensing Subsystem	Commands/interaction with HCI
Video (Indoor)	Cabin Passengers	User's social life Commands/interaction with HCI
LiDAR	Cabin Passengers	User's social life Commands/interaction with HCI
AMS-HCI Response	Autonomous Motion Subsystem	Response to HCI-AMS Command
Inter HCI Information	Remote HCI	HCI-to-HCI information
Output data	**To**	**Comments**
Full Environment Representation Audio	Passenger Cabin	For passengers to hear external Environment
Full Environment Representation Video	Passenger Cabin	For passengers to view external Environment
Inter HCI Information	Remote HCI	HCI-to-HCI information
HCI-AMS Command	Autonomous Motion Subsystem	HCI-to-AMS information
Machine Portable Avatar	Cabin Passengers	HCI's avatar.

5.6 Environment Sensing Subsystem in a Connected Autonomous Vehicle

5.7 Functions of Environment Sensing Subsystem

The Environment Sensing Subsystem (ESS) of a Connected Autonomous Vehicle (CAV):

1. Uses all Subsystem devices to acquire as much as possible information from the Environment as electromagnetic and acoustic data.
2. Receives an initial estimate of the Ego CAV's Spatial Attitude generated by the Motion Actuation Subsystem
3. Receives Environment Data (e.g., temperature, pressure, humidity, etc.) from the Motion Actuation Subsystem.
4. Produces a sequence of Basic Environment Representations (BER) for the journey.

5. Passes the Basic Environment Representations to the Autonomous Motion Subsystem.

5.8 Reference Architecture of Environment Sensing Subsystem

Figure 4 gives the Environment Sensing Subsystem Reference Model.

The typical sequence of operations of the Environment Sensing Subsystem is:
1. Compute the CAV's Spatial Attitude using the initial Spatial Attitude provided by the Motion Actuation Subsystem and the GNSS.
2. Receives Environment Sensing Technology (EST)-specific Data, e.g., RADAR Data provided by the RADAR EST.
3. Produce and send EST-specific Alert, if necessary, to Autonomous Motion Subsystem.
4. Access the Basic Environment Representation at a previous time if necessary.
5. Produce EST-specific Scene Descriptors, e.g., the RADAR Scene Descriptors.
6. Integrate the Scene Descriptors from different ESTs into the Basic Environment Representation.

Note that *Figure 4* assumes that:
1. Traffic Signalisation Recognition produces the Road Topology by analysing Camera Data. The model of *Figure 4* can easily be extended to the case where Data from other ESTs is processed to compute or help compute the Road Topology.
2. Environment Sensing Technologies are individually processed. An implementation may produce a single Scene Descriptors from two or more ESTs.

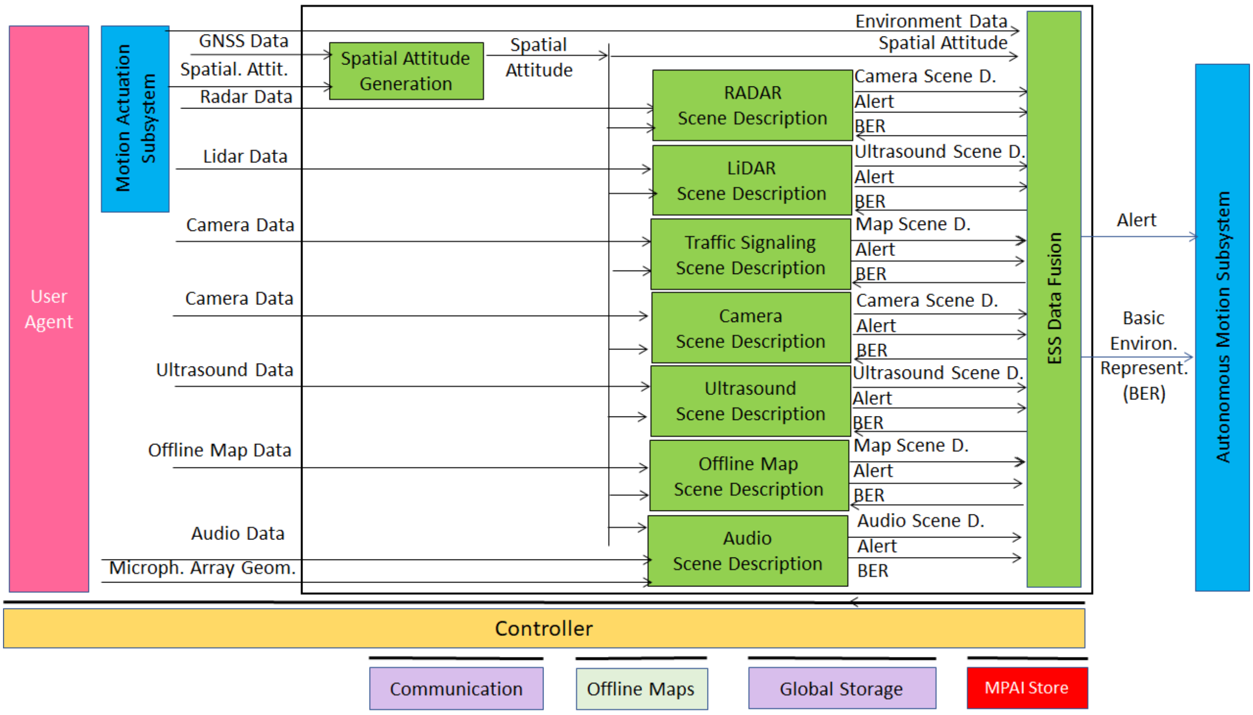

Figure 4 – Environment Sensing Subsystem Reference Model

5.9 I/O Data of Environment Sensing Subsystem

The currently considered Environment Sensing Technologies (EST) are:
1. Global navigation satellite system or GNSS (~1 & 1.5 GHz Radio).
2. Geographical Position and Orientation, and their time derivatives up to 2^{nd} order (Spatial Attitude).

3. Visual Data in the visible range, possibly supplemented by depth information (400 to 700 THz).
4. LiDAR Data (~200 THz infrared).
5. RADAR Data (~25 & 75 GHz).
6. Ultrasound Data (> 20 kHz).
7. Audio Data in the audible range (16 Hz to 20 kHz).
8. Spatial Attitude (from the Motion Actuation Subsystem).
9. Other environmental data (temperature, humidity, ...).

Offline Map data can be accessed either from stored information or online.

Table 4 gives the input/output data of the Environment Sensing Subsystem.

Table 4 – I/O data of Environment Sensing Subsystem

Input data	**From**	**Comment**
Radar Data	~25 & 75 GHz Radio	Capture Environment with Radar
Lidar Data	~200 THz infrared	Capture Environment with Lidar
Camera Data	Video (400-800 THz)	Capture Environment with Cameras
Ultrasound Data	Audio (>20 kHz)	Capture Environment with Ultrasound
Offline Map Data	Local storage or online	cm-level data at time of capture
Audio Data	Audio (16 Hz-20 kHz)	Capture Environment or cabin with Microphone Array
Microphone Array Geometry	Microphone Array	Microphone Array disposition
Global Navigation Satellite System (GNSS) Data	~1 & 1.5 GHz Radio	Get Pose from GNSS
Spatial Attitude	Motion Actuation Subsystem	To be fused with GNSS data
Other Environment Data	Motion Actuation Subsystem	Temperature etc. added to Basic Environment Representation
Output data	**To**	**Comment**
Alert	Autonomous Motion Subsystem	Critical information from an EST.
Basic Environment Representation	Autonomous Motion Subsystem	ESS-derived representation of external Environment

5.10 Autonomous Motion Subsystem in a Connected Autonomous Vehicle

5.11 Functions of Autonomous Motion Subsystem

The functions of the Autonomous Motion Subsystem (AMS) are:
1. Receive a request to reach a destination as instructed by Human-CAV Interaction (HCI).
2. Request current Pose to Environment Sensing Subsystem (ESS).
3. Converse with HCI and settle on final Route.
4. Receive Basic Environment Representation (BER) from ESS.
5. Broadcast appropriate BER subsets to Remote AMSs.
6. Respond to specific Remote AMS requests.
7. Produce Full Environment Representation.

8. Generate Paths (Plath Planner).
9. Generate Goal and Trajectory (Motion Planner).
10. Check whether Trajectory can be implemented (Obstacle Avoider).
11. Issue Command to Motion Actuation Subsystem.

5.12 Reference Architecture of Autonomous Motion Subsystem

Figure 5 gives the Autonomous Motion Subsystem Reference Model.

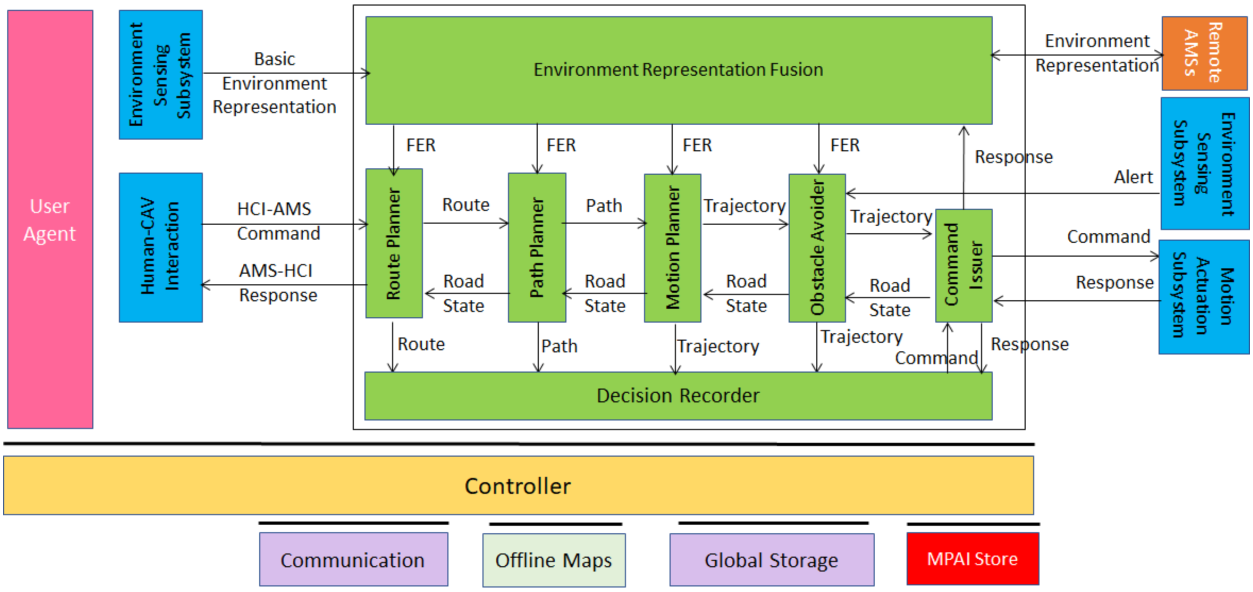

Figure 5 – Autonomous Motion Subsystem Reference Model

This is the operation of the Reference Model:

1. A human requests the Human-CAV Interaction to take them to a destination.
2. HCI interprets request and passes interpretation to the AMS.
3. The AMS activates the Route Planner to generate a set of Waypoints starting from the current Pose, obtained from the Full Environment Representation, up to the destination.
4. The Waypoints enter the Path Planner which generates a set of Poses to reach the next Waypoint.
5. For each Path, the Motion Planner generates a Trajectory to reach the next Pose.
6. Obstacle Avoider receives the Trajectory and checks if an Alert should was received.
7. If an Alert was received, Obstacle Avoider checks whether the implementation of the Trajectory creates a collision.
    a. If a collision is indeed detected, Obstacle Avoider requests a new Trajectory from the Motion Planner.
    b. If no collision is detected, Obstacle Avoider issues a Command to Motion Actuation Subsystem.
8. The Motion Actuation Subsystem sends MAS-AMS Response about the execution of the Command.
9. The AMS, based on the MAS-AMS Responses received potentially conveying changes in the Environment, can decide to discontinue the execution of the earlier Command and issue another AMS-MAS Command instead.
10. The decision of each element of the said chain may be recorded in the Decision Recorder ("black box").

5.13 I/O Data of Autonomous Motion Subsystem

Table 5 gives the input/output data of Autonomous Motion Subsystem.

Table 5 – I/O data of Autonomous Motion Subsystem

Input data	**From**	**Comment**
Basic Environment Representation	Environment Sensing Subsystem	CAV's Environment representation.
Alert	Environment Sensing Subsystem	Critical information from an EST in ESS.
HCI-AMS Command	Human-CAV Interaction	Human commands, e.g., "take me home"
Environment Representation	Remote AMSs	Other CAVs and vehicles, and roadside units.
MAS-AMS Response	Motion Actuation Subsystem	CAV's response to AMS-MAS Command.
Output data	**To**	**Comment**
AMS-HCI Response	Human-CAV Interaction	MAS's response to AMS-MAS Command
AMS-MAS Command	Motion Actuation Subsystem	Macro-instructions, e.g., "in 5s assume a given Spatial Attitude".
Environment Representation	Remote AMSs	For information to other CAVs

5.14 Avatar-Based Videoconference – Transmitting Client

Avatar-Based Videoconference is a videoconference whose participants are avatars realistically impersonating human participants. See Chapter 5 of Annex 1 - MPAI Basics for more information on the Avatar-Based Videoconference Use Case. This is fully specified in [9].

5.14.1 Functions of Transmitting Client

The function of a Transmitting Client is to:

1. Receive from a Participant:
1.1. Input Audio from the microphone.
1.2. Input Video from the camera.
1.3. Participant's Avatar Model.
1.4. Participant's language preferences (e.g., EN-US, IT-CH).
2. Send to the Server:
2.1. Speech Object (for Authentication).
2.2. Face Object (for Authentication).
2.3. Input Portable Avatars containing:
2.3.1. Language preferences (at the start).
2.3.2. Avatar Model (at the start).
2.3.3. Speech.
2.3.4. Avatar Descriptors.

5.14.2 Reference Architecture of Transmitting Client

Figure 6 gives the architecture of Transmitting Client AIW. Red text refers to data sent at meeting start.

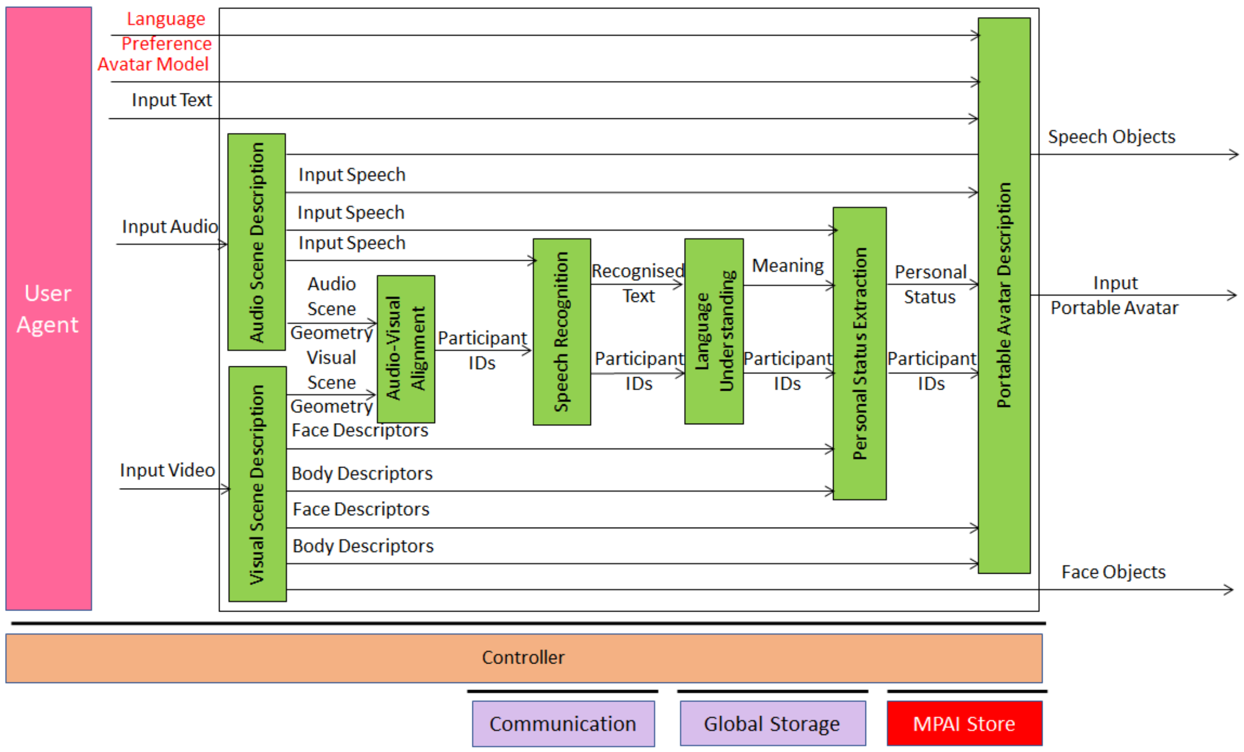

Figure 6 – Reference Model of Avatar Videoconference Transmitting Client

At the start, each participant sends to the Server:
1. Language preferences
2. Avatar model.
3. Speech Object (for Authentication).
4. Face Object (for Authentication).

During the meeting
1. The following AIMs of the Transmitting Clients produce:
1.1. Audio Scene Description: Audio Scene Descriptors.
1.2. Visual Scene Description: Visual Scene Descriptors.
1.3. Audio-Visual Alignment: Identifiers of Audio, Visual, and Audio-Visual Descriptors.
1.4. Speech Recognition: Recognised Text.
1.5. Face Description: Face Descriptors.
1.6. Body Description: Body Descriptors.
1.7. Personal Status Extraction: Personal Status.
1.8. Language Understanding: Meaning.
1.9. Portable Avatar Description: Avatar Descriptors.
2. The Transmitting Clients send Portable Avatar to the Server for distribution to Receiving Clients:

5.14.3 Input and output data of Transmitting Client

Table 6 gives the input and output data of the Transmitting Client AIW:

Table 6 – Input and output data of Client Transmitting AIW

Input	Comments
Input Text	Chat text used by a human to communicate with Virtual Secretary or other participants

Language Preference	The language participant wishes to speak and hear at the videoconference.
Input Audio	Audio of Speech of participants in a meeting room.
Input Video	Video of participants in a meeting room.
Avatar Model	The avatar model selected by the participant.
Output	**Comments**
Input Portable Avatar	Portable Avatar produced by Transmitting Client.
Speech Object	For authentication by Server.
Face Object	For authentication by Server.

5.15 Avatar-Based Videoconference – Server

Avatar-Based Videoconference is a videoconference whose participants are avatars realistically impersonating human participants. See Chapter 5 of Annex 1 - MPAI Basics for more information on the Avatar-Based Videoconference Use Case. This is fully specified in [9].

5.15.1 Functions of Server

The Server:

1. *At the start*:
1.1. Receives Speech Object and Speech Objects of each Participant.
1.2. Authenticates Participants.
1.3. Receives Portable Avatars each containing Language Preference and Avatar Model.
1.4. Selects a Visual Environment.
1.5. Selects the Spatial Attitudes of Avatar Models.
1.6. Selects the common meeting language.
1.7. Distributes all Portable Avatars each containing: Visual Environment, Language Preference, Avatar Model, and Spatial Attitude.
2. *During the videoconference:*
2.1. Receives Participants' and Virtual Secretary's Avatar Descriptors.
2.2. Translates participants' Speech according to their language preferences.
2.3. Sends Portable Avatars containing Avatar ID, Text, Speech translated to the common meeting language, Face Descriptors and Gesture Descriptors to Virtual Secretary.
2.4. Receives Virtual Secretary's Portable Avatar containing Avatar ID, Text, Speech in the common meeting language, Face Descriptors and Gesture Descriptors.
2.5. Translates Virtual Secretary's Speech according to each participant's language preferences.
2.6. Sends Participants' and Virtual Secretary's Portable Avatars containing Avatar ID, Text, Translated Speech, Face Descriptors and Gesture Descriptors to Receiving Clients.

5.15.2 Reference Architecture of Server

Figure 5 gives the architecture of Server AIW. Red text refers to data sent at meeting start.

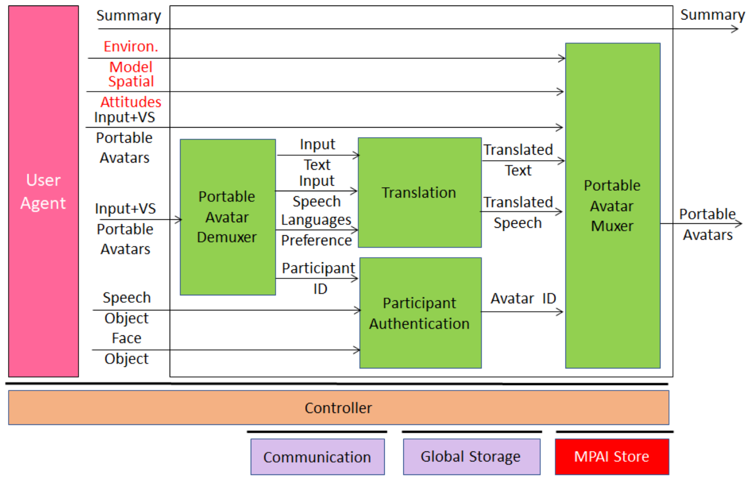

Figure 7 – Reference Model of Avatar-Based Videoconference Server

5.15.3 I/O Data of Server

Table 7 gives the input and output data of Server AIW.

Table 7 – Input and output data of Server AIW

Input	Comments
Summary	From Virtual Secretary.
Visual Environment Model	Set by Server.
Spatial Attitudes	Set by Server.
Input+VS Portable Avatars	From Transmitting Clients and Virtual Secretary
Speech Objects	Participants' Speech Object for Authentication.
Face Objects	Participants' Face Object for Authentication.
Outputs	**Comments**
Summary	As above.
Portable Avatars	As re-multiplexed by Server.

5.16 Avatar-Based Videoconference – Receiving Client

Participants in Avatar-Based Videoconference are avatars realistically impersonating human participants at remote locations. See Chapter 5 of Annex 1 - MPAI Basics for more information on the Avatar-Based Videoconference Use Case, fully specified in [7].

5.16.1 Functions of Receiving Client

The Function of the Client (Receiving Side) is to:

1. Create the Environment using the Environment Model.
2. Place and animate the Avatar Models at their Spatial Attitudes.
3. Add Speech to Avatar's mouth.
4. Render the Audio-Visual Scene as seen from the Participant-selected Point of View.

5.16.2 Reference Architecture of Receiving Client

The Receiving Client:

1. At the start

1.1. Receives the Visual Environment and the Portable Avatars containing:

1.1.1. The Visual Environment.

1.1.2. The Avatar Models

1.1.3. The Spatial Attitudes
1.2. Creates the initial AV Scene.
2. During the Videoconference:
2.1. Receives the Avatar Models containing:
2.1.1. Speech
2.1.2. Body Descriptors
2.1.3. Face Descriptors
2.2. Creates the running AV Scene using each Avatar's:
2.3. The Body and Face Descriptors.
2.4. The Speech.
3. Renders the Audio-Visual Scene based on the selected Point of View.

Figure 6 gives the architecture of Client Receiving AIW. Red text refers to data received at the meeting start.

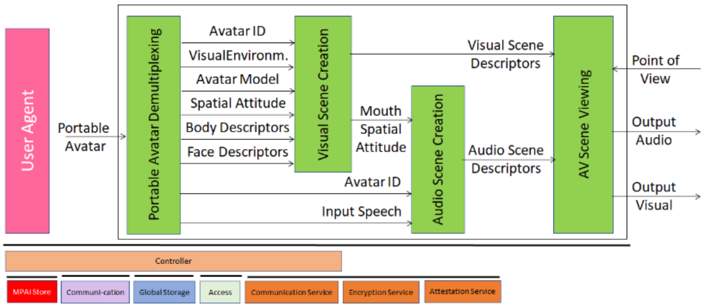

Figure 8 – Reference Model of Avatar-Based Videoconference Client (Receiving Side)

Notes:
1. An implementation may decide to display text with a visual image for accessibility purposes.
2. Audio Environment is added for completeness. However, This Standard does not provide a specification for it.

5.16.3 I/O Data of Receiving Client

Table 8 gives the input and output data of Receiving Client.

Table 8 – Input and output data of Receiving Client AIW

Input	Comments
Point of View	Participant-selected point of view to see visual objects and hear audio objects in the Virtual Environment.
Portable Avatars	Portable Avatars from Server.
Output	**Comments**
Output Audio	Presented using loudspeaker (array)/earphones.
Output Visual	Presented using 2D or 3D display.

6 Composite AI Modules

6.1 Visual Spatial Object Identification (OSD-VOI)

6.1.1 Functions of Visual Spatial Object Identification

The purpose of the Visual Spatial Object Identification (OSD-VOI) AIM is to provide the Identifier of a Physical Object in an Environment with a plurality of Objects that a human indicates by pointing at it with a finger.

6.1.2 Reference Architecture of Visual Spatial Object Identification

Figure 9 depicts the AIM implementing the Spatial Object Identification AIM.

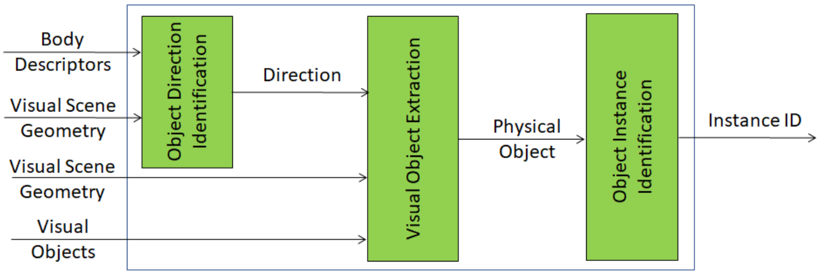

Figure 9 – Reference Model of the Visual Object Identification AIM

The workflow of Visual Spatial Object Identification unfolds as follows:

1. Direction Identification provides the (ϕ,θ) angles obtained by analysing the finger of the human.
2. Object Extraction uses the Visual Scene Geometry and the Direction to find the Object intersected by the line identified by (ϕ,θ) passing through the finger. It is assumed that one and only one Object is found.
3. Object Instance Identification provides the ID of the Object Instance.

6.1.3 Input/output data of Visual Spatial Object Identification

Table 9 gives the input/output data of Spatial Object Identification.

Table 9 – I/O data of Spatial Object Identification

Input data	**From**	**Comment**
Body Descriptors	Visual Scene Description	There is a human pointing to an object
Physical Objects	Visual Scene Description	There are many scene objects
Scene Geometry	Visual Scene Description	Full description of the scene
Output data	**To**	**Comments**
Physical Object Instance ID	Human or another AIM	Human points to one object only

6.1.4 SubAIMs

The Visual Spatial Object Identification Composite AIMs includes the following SubAIMs:

1. Direction Identification
2. Visual Object Extraction
3. Object Instance Identification.

6.1.4.1 *Visual Direction Identification*

6.1.4.1.1 Function

Visual Direction Identification (VOI-VDI):

1. Receives Visual Scene Geometry and Body Descriptors.
2. Produces the direction of a line traversing the forefinger of the Entity.

6.1.4.1.2 Reference Architecture

Figure 10 depicts the Reference Architecture of the Visual Direction Identification AIM.

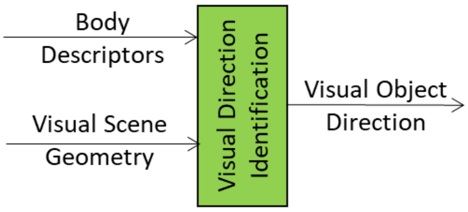

Figure 10 – The Visual Direction Identification AIM

6.1.4.1.3 I/O Data

Table 10 specifies the Input and Output Data of the Visual Direction Identification AIM.

Table 10 – I/O Data of the Visual Direction Identification AIM

Input	Description
Body Descriptors	The Descriptors of the Body Objects of Entities in the Visual Scene.
Visual Scene Geometry	The digital representation of the spatial arrangement of the Visual Objects of the Scene.
Output	**Description**
Visual Object Direction	The direction of the line traversing the forefinger of the target Entity.

6.1.4.2 *Visual Object Extraction*

6.1.4.2.1 Function

Visual Object Extraction (VOI-VOE):

1. Receives Visual Scene Geometry, Visual Objects, and Direction.
2. Singles out the Visual Object indicated by the Entity.

6.1.4.2.2 Reference Architecture

Figure 11 depicts the Reference Architecture of the Visual Object Extraction AIM.

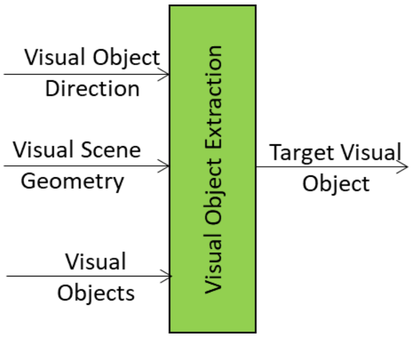

Figure 11 – The Visual Object Extraction AIM

6.1.4.2.3 I/O Data

Table 11 specifies the Input and Output Data of the Visual Object Extraction AIM.

Table 11 – I/O Data of the Visual Object Extraction AIM

Input	**Description**
Visual Object Direction	The direction of the line traversing the forefinger of the Entity.
Visual Scene Geometry	The digital representation of the spatial arrangement of the Visual Objects of the Scene.
Visual Objects	The Visual Objects identified in the Visual Scene Geometry.
Output	**Description**
Target Visual Object	The Visual Object crossed by the line traversing the forefinger of the Entity.

6.1.4.3 Object Instance Identification

6.1.4.3.1 Function

Object Instance Identification (VOI-OII):
1. Receives a Visual Object.
2. Produces an Instance ID identifying an element of a set of Visual Objects belonging to a level in a taxonomy.

6.1.4.3.2 Reference Architecture

Figure 12 depicts the Reference Architecture of the Object Instance Identification AIM.

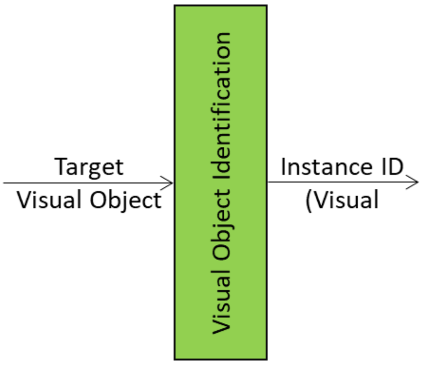

Figure 12 – The Visual Object Identification AIM

6.1.4.3.3 I/O Data

Table 12 specifies the Input and Output Data of the Object Instance Identification AIM.

Table 12 – I/O Data of Object Instance Identification

Input	**Description**
Target Visual Object	The Visual Object crossed by the line traversing the forefinger of the Entity.
Output	**Description**
Visual Instance ID	The Identifier of the specific Visual Object belonging to a level in the taxonomy.

6.1.4.4 JSON Metadata of Visual Spatial Object Identification

Specified in Annex 5 - Visual Spatial Object Identification (OSD-VOI).

6.2 Audio-Visual Scene Description (OSD-AVD)

6.2.1 Scope

The Audio-Visual Scene Description (OSD-AVD) Composite AIM receives two independently developed Audio Scene Descriptors and Visual Scene Descriptors in the same Virtual Space and produces Audio-Visual Scene Descriptors whose co-located Audio Objects and Visual Objects have the same or related identifiers.

6.2.2 Reference Architecture

Figure 13 gives the Reference Model of Audio-Visual Scene Description.

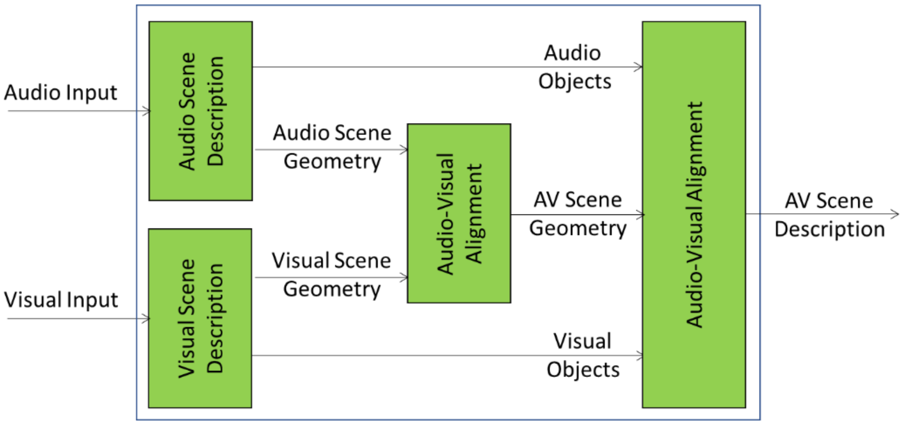

Figure 13 - Reference Model of Audio-Visual Scene Description

6.2.3 Input/output data

*Table 13*gives the input/output data of Audio-Visual Scene Description.

Table 13 – I/O data of Audio-Visual Scene Description

Input data	From	Comment
Input Audio	A real environment	The Input Audio and Input Visual are from the same scene
Input Visual	A real environment	The Input Audio and Input Visual are from the same scene
Output data	**To**	**Comments**
AV Scene Descriptors	Downstream AIM	The co-located Audio and Visual Objects in the Scene have the same or related identifi-ers.

6.2.4 SubAIMs

The Audio-Visual Scene Description Composite AIMs includes the following SubAIMs:

1. Audio Scene Description
2. Visual Scene Description
3. Audio-Visual Alignment.

6.2.4.1 Audio Scene Description

Specified in MPAI-CAE V2.1 [3].

6.2.4.2 Visual Scene Description

6.2.4.2.1 Scope

The scope of the Visual Scene Description Composite AIM is to:

1. Capture the Input Visual
2. Provide the following output:

2.1. Visual Objects
2.2. Scene Geometry
2.3. Scene Descriptors

6.2.4.2.2 Reference Architecture

Figure 14 depicts the AIM implementing the Visual Scene Description AIM.

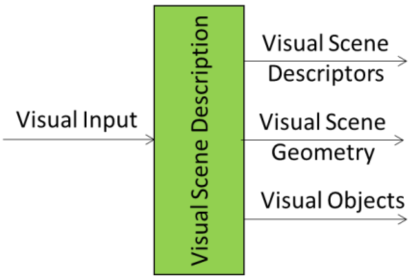

Figure 14 - Visual Scene Description AIM

6.2.4.2.3 Input/Output Data

Table 14 gives the input/output data of Spatial Object Identification.

Table 14 – I/O data of Audio Scene Description

Input data	From	Comment
Input Visual	A real environment.	The environment includes objects in scenes.
Output data	**To**	**Comments**
Visual Scene Descriptors	Downstream AIM	A Data Type including the digital representation of the visual features of a digital scene.
Visual Scene Geometry	Downstream AIM	Interpreted Face Descriptors
Visual Objects	Downstream AIM	Visual Objects belong to two types of Objects: 1. Digitised Humans [7] represented by: 1.1. Body Descriptors 1.2. Scene Descriptors 2. Generic Visual Onjects

6.2.4.3 Audio-Visual Alignment (OSD-AVA)

6.2.4.3.1 Scope

The Audio-Visual Alignment Composite AIM takes the Objects of two independently developed Audio Scene Description and Visual Scene Descriptions in the same Virtual Space, gives related identifiers to the Audio Objects and Visual Objects that have the same location in the Virtual Space, and gives independent identifiers to independently located Audio and Visual Objects.

6.2.4.3.2 Reference Architecture

Figure 15 gives the Reference model of Audio-Visual Alignment.

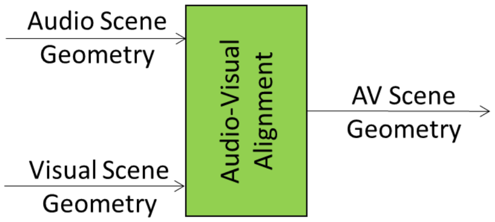

Figure 15 - Reference Model of Audio-Visual Alignment

6.2.4.3.3 Input/Output Data

Figure 14 gives the input/output data of Spatial Object Identification.

Table 15 – I/O data of Audio-Visual Alignment

Input data	**From**	**Comment**
Audio Scene Geometry	Another AIM	A Data Type describing the Spatial arrangement of the Audio Objects of a Scene.
Visual Scene Geometry	Another AIM	A Data Type describing the Spatial arrangement of the Audio Objects of a Scene.
Output data	**To**	**Comments**
Audio-Visual Scene Geometry	Downstream AIM	The identifiers of the co-located Audio and Visual Objects have the same or related identifiers

7 Data Formats

Table 16 provides the list of Data Formats target of the Call for Technologies.

Table 16 – Data formats

Name of Data Format	**Subsection**	**Use Case**
Coordinates, Angles, and Objects	7.1	ARA-ABV
		MMC-CAS
		MMC-HCI
		MPAI-CAV
		MPAI-MMM
Spatial Attitude	7.2	ARA-ABV
		MMC-CAS
		MMC-HCI
		MPAI-CAV
		MPAI-MMM
Audio Scene Geometry	7.3	ARA-ABV
		MMC-HCI
		MPAI-CAV
		MPAI-MMM
Audio Scene Descriptors	7.4	ARA-ABV
		MMC-HCI
		MPAI-CAV

		MPAI-MMM
Visual Scene Geometry	7.5	ARA-ABV
		MMC-CAS
		MMC-HCI
		MPAI-CAV
		MPAI-MMM
		MMC-HCI
Visual Scene Descriptors	7.6	ARA-ABV
		MMC-CAS
		MMC-HCI
		MPAI-CAV
		MPAI-MMM
		MMC-HCI
Audio-Visual Scene Geometry	7.7	ARA-ABV
		MMC-CAS
		MMC-HCI
		MPAI-CAV
		MPAI-MMM
		MMC-HCI
Audio-Visual Scene Descriptors	7.8	ARA-ABV
		MMC-CAS
		MMC-HCI
		MPAI-CAV
		MPAI-MMM
		MMC-HCI

The following Sections specify of the data formats.

7.1 Coordinates, Angles, and Objects

Figure 16 depict the regular way of defining Cartesian. *Figure 17* depicts the Cartesian Coordinates applicable to a visual capture device such as camera or LiDAR placed in an Environment with the (x,y) plane perpendicular and crossing the Device's sensors. The z axis is perpendicular to the (x,y) plane and pointing to the captured scene.

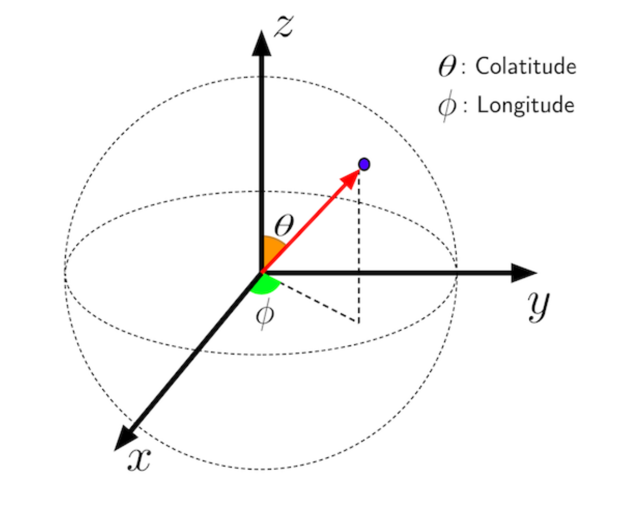

Figure 16 – Cartesian and Spherical Coordinates

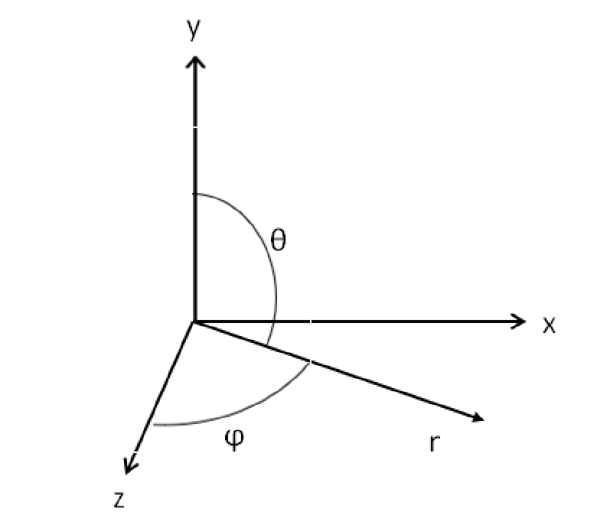

Figure 17 – Cartesian Coordinates of a capture device

Figure 18, *Figure 19*, and *Figure 20* graphically represent how different applications associate the local (x,y,z) coordinates with the roll, pitch, and yaw rotations.

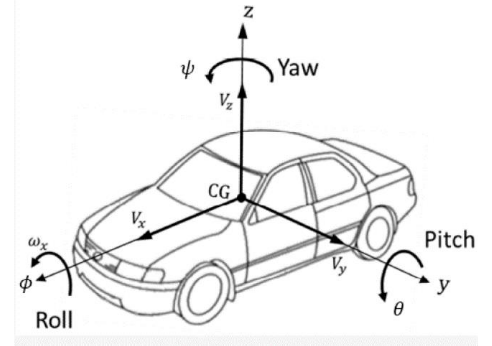

Figure 18 – Car

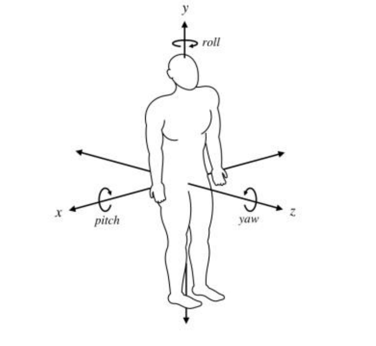

Figure 19 – Human

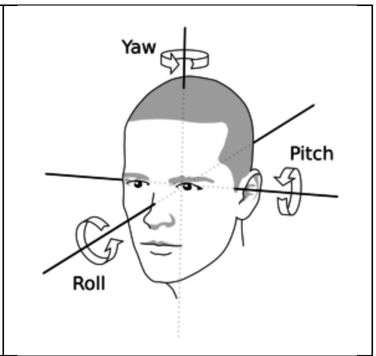

Figure 20 – Head

7.2 Spatial Attitude

Table 17 gives the components of the Spatial Attitude of an Object. The Position of an Object is that of a representative point in the Object.

7.2.1 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Object Spatial Attitude",
  "type": "object",
  "properties": {
    "Header": {
      "type": "object",
      "properties": {
        "Standard": {
          "type": "string"
        },
        "Version": {
          "type": "integer"
        },
        "Subversion": {
          "type": "integer"
        }
      }
```

```
},
"OSAID": {
  "type": "string"
},
"General": {
  "type": "object",
  "properties": {
    "CoordType": {
      "type": "number"
    },
    "ObjectType": {
      "type": "number"
    },
    "Precision": {
      "type": "number"
    },
    "MediaType": {
      "type": "number"
    }
  }
},
"CartPosition": {
  "type": "array",
  "minItems": 3,
  "maxItems": 3,
  "items": {
    "type": "number"
  }
},
"SpherPosition": {
  "type": "array",
  "minItems": 3,
  "maxItems": 3,
  "items": {
    "type": "number"
  }
},
"Orientation": {
  "type": "array",
  "minItems": 3,
  "maxItems": 3,
  "items": {
    "type": "number"
  }
},
"CartVelociry": {
  "type": "array",
  "minItems": 3,
  "maxItems": 3,
  "items": {
    "type": "number"
  }
},
"SpherVelocity": {
  "type": "array",
  "minItems": 3,
  "maxItems": 3,
  "items": {
    "type": "number"
  }
},
"OrientVelocity": {
  "type": "array",
  "minItems": 3,
  "maxItems": 3,
  "items": {
    "type": "number"
  }
},
"CartAccel": {
  "type": "array",
  "minItems": 3,
  "maxItems": 3,
  "items": {
```

```
          "type": "number"
        }
      },
      "SpherAccel": {
        "type": "array",
        "minItems": 3,
        "maxItems": 3,
        "items": {
          "type": "number"
        }
      },
      "OrientAccel": {
        "type": "array",
        "minItems": 3,
        "maxItems": 3,
        "items": {
          "type": "number"
        }
      }
    }
  }
}
```

7.2.2 Semantics

Table 17 provides the semantics of the components of the Spatial Attitude. The following should be noted:

1. The first byte is always present.
2. Each of the other components is optional.
3. Each of Position, Velocity, and Acceleration is provided either in Cartesian (X,Y,Z) or Spherical (r,φ,θ) Coordinates.
4. The Euler angles are indicated by (α,β,γ).

Table 17 – Components of the Spatial Attitude

<table>
<tr><td>HEADER</td><td>9 Bytes</td><td></td></tr>
<tr><td>• Standard</td><td>7 Bytes</td><td>The string OSD-OSA</td></tr>
<tr><td>• Version</td><td>1 Byte</td><td>Major version</td></tr>
<tr><td>• Subversion</td><td>1 Byte</td><td>Minor version</td></tr>
<tr><td>OSAID</td><td>16 Bytes</td><td>UUID Identifier of Object Spatial Attitude.</td></tr>
<tr><td>General</td><td></td><td></td></tr>
<tr><td>• CoordType</td><td>bit 0</td><td>0: Cartesian, 1: Spherical</td></tr>
<tr><td>• ObjectType</td><td>bit 1-2</td><td>00: Digital Human
01: Generic
10 and 11: reserved</td></tr>
<tr><td>• Precision</td><td>bit 3</td><td>0: single precision; 1: double precision</td></tr>
<tr><td>• MediaType</td><td>bit 4-6</td><td>000: Audio; 001: Visual; 010: Haptic; 011: Smell; 100: RADAR; 101: LiDAR; 110: Ultrasound; 111: reserved</td></tr>
<tr><td>• Reserved</td><td>bit 6-7</td><td>reserved</td></tr>
<tr><td>• SpatialAttitudeMask</td><td>2 Bytes</td><td>3*3 matrix of booleans (by rows)<table><tr><td></td><td>Position</td><td>Velocity</td><td>Acceleration</td></tr><tr><td>Cartesian</td><td></td><td></td><td></td></tr><tr><td>Spherical</td><td></td><td></td><td></td></tr><tr><td>Orientat.</td><td></td><td></td><td></td></tr></table></td></tr>
<tr><td colspan="3">Position and Orientation</td></tr>
<tr><td>• CartPosition (X,Y,Z)</td><td>12/24 Bytes</td><td>Array (in metres)</td></tr>
<tr><td>• SpherPosition (r,φ,θ)</td><td>12/24 Bytes</td><td>Array (in metres and degrees)</td></tr>
</table>

• Orient (α,β,γ)	12/24 Bytes	Array (in degrees)
Velocity of Position and Orientation		
• CartVelocity (X,Y,Z)	12/24 Bytes	Array (in metres)
• SpherVelocity (r,φ,θ)	12/24 Bytes	Array (in metres and degrees)
• OrientVelocity (α,β,γ)	12/24 Bytes	Array (in degrees)
Acceleration of Position and Orientation		
• CartAccel (X,Y,Z)	12/24 Bytes	Array (in metres)
• SpherAccel (r,φ,θ)	12/24 Bytes	Array (in metres and degrees)
• OrientAccel (α,β,γ)	12/24 Bytes	Array (in degrees)

7.3 Audio Scene Geometry

The Audio Scene Geometry format is specified in [3]. It is reported here for convenience.

7.3.1 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Audio Scene Geometry",
  "type": "object",
  "properties": {
    "Header": {
      "type": "object",
      "properties": {
        "Standard": {
          "type": "string"
        },
        "Version": {
          "type": "integer"
        },
        "Subversion": {
          "type": "integer"
        }
      }
    },
    "ASDID": {
      "type": "string"
    },
    "Time": {
      "type": "object",
      "properties": {
        "TimeType": {
          "type": "boolean"
        },
        "StartTime": {
          "type": "number"
        },
        "EndTime": {
          "type": "number"
        }
      }
    },
    "BlockSize": {
      "type": "integer"
    },
    "AudioObjectCount": {
      "type": "integer"
    },
    "AudioObjectsData": {
      "type": "object",
      "properties": {
        "AudioObjectID": {
          "type": "string"
        },
        "SpatialAttitude": {
          "$ref": "https://schemas.mpai.community/OSD/V1.0/data/SpatialAttitude.json"
```

```
          }
        }
      }
    }
  }
```

7.3.2 Semantics

Table 18 provides the semantics of the Audio Scene Geometry.

Table 18 – Audio Scene Geometry Semantics

Label	Size	Description
HEADER	9 Bytes	
• Standard	7 Bytes	The string CAE-ASD
• Version	1 Byte	Major version
• Subversion	1 Byte	Minor version
ASDID	16 Bytes	UUID Identifier of Audio Scene Descriptors set.
Time	17 Bytes	Collects various data expressed with bits
• TimeType	0 bit	0=Relative: time starts at 0000/00/00T00:00 1=Absolute: time starts at 1970/01/01T00:00.
• Reserved	1-7 bits	reserved
• StartTime	8 Bytes	Start of current Audio Scene Descriptors (in μs).
• EndTime	8 Bytes	End of current Audio Scene Descriptors (in μs).
BlockSize	4 Bytes	Minimum BlockSize: ≥ 256.
AudioObjectCount	1 Byte	Number of Audio Objects in the Audio Scene.
AudioObjectsData	N1 Bytes	Data associated to each Audio Object.
• AudioObjectID	1 Byte	ID of a specific Audio Object in the Audio Scene.
• SamplingRate	0-3 bits	0:8, 1:16, 2:24, 3:32, 4:44.1, 5:48, 6: 64, 7: 96, 8: 192 (all kHz)
• SampleType	4-5 bits	0:16, 1:24, 2:32, 3:64 (all bits/sample)
• Reserved	6-7 bits	
• Spatial Attitude	N2 Bytes	

7.4 Audio Scene Descriptors

The Audio Scene Descriptors format is specified in [3]. It is reported here for convenience.

7.4.1 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Audio Scene Descriptors",
  "type": "object",
  "properties": {
    "Header": {
      "type": "object",
      "properties": {
        "Standard": {
          "type": "string"
        },
        "Version": {
          "type": "integer"
        },
        "Subversion": {
          "type": "integer"
        }
      }
    },
    "ASDID": {
```

```
        "type": "string"
      },
      "Time": {
        "type": "object",
        "properties": {
          "TimeType": {
            "type": "boolean"
          },
          "StartTime": {
            "type": "number"
          },
          "EndTime": {
            "type": "number"
          }
        }
      },
      "BlockSize": {
        "type": "integer"
      },
      "AudioObjectCount": {
        "type": "integer"
      },
      "AudioObjectsData": {
        "type": "object",
        "properties": {
          "AudioObjectID": {
            "type": "string"
          },
          "SamplingRate": {
            "type": "number"
          },
          "SamplingType": {
            "type": "number"
          },
          "SpatialAttitude": {
            "$ref": "https://schemas.mpai.community/OSD/V1.0/data/SpatialAttitude.json"
          },
          "AudioObject": {
            "type": "object",
            "properties": {
              "FormatID": {
                "type": "integer"
              },
              "ObjectLength": {
                "type": "integer"
              },
              "DataInObject": {
                "$ref": "https://schemas.mpai.community/CAE/V2.1/data/AudioObject.json"
              }
            }
          }
        }
      }
    }
  }
}
```

7.4.2 Semantics

Table 19 provides the semantics of Audio Scene Descriptors.

Table 19 – Audio Scene Descriptors

Label	Size	Description
HEADER	9 Bytes	
• Standard	7 Bytes	The string CAE-ASD
• Version	1 Byte	Major version
• Subversion	1 Byte	Minor
ASDID	16 Bytes	UUID Identifier of Audio Scene Descriptors set.

Time	17 Bytes	Collects various data expressed with bits
• TimeType	0 bit	0=Relative: time starts at 0000/00/00T00:00 1=Absolute: time starts at 1970/01/01T00:00.
• Reserved	1-7 bits	reserved
• StartTime	8 Bytes	Start of current Audio Scene Descriptors (in μs).
• EndTime	8 Bytes	End of current Audio Scene Descriptors (in μs).
BlockSize	4 Bytes	Minimum BlockSize: ≥ 256.
AudioObjectCount	1 Byte	Number of Audio Objects in the Audio Scene.
AudioObjectsData	N1 Bytes	Data associated to each Audio Object.
• AudioObjectID	1 Byte	ID of a specific Audio Object in the Audio Scene.
• SamplingRate	0-3 bits	0:8, 1:16, 2:24, 3:32, 4:44.1, 5:48, 6: 64, 7: 96, 8: 192 (all kHz)
• SampleType	4-5 bits	0:16, 1:24, 2:32, 3:64 (all bits/sample)
• Reserved	6-7 bits	
• Spatial Attitude	N2 Bytes	According to MPAI-OSD V1
• AudioObject	N3 Bytes	
○ FormatID	1 Byte	Audio Object Format Identifier
○ ObjectLength	4 Bytes	Number of Bytes in Audio Object
○ DataInObject	N4 Bytes	Data of Audio Object

7.5 Visual Scene Geometry

7.5.1 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Visual Scene Geometry",
  "type": "object",
  "properties": {
    "Header": {
      "type": "object",
      "properties": {
        "Standard": {
          "type": "string"
        },
        "Version": {
          "type": "integer"
        },
        "Subversion": {
          "type": "integer"
        }
      }
    },
    "VSDID": {
      "type": "string"
    },
    "Time": {
      "type": "object",
      "properties": {
        "TimeType": {
          "type": "boolean"
        },
        "StartTime": {
          "type": "number"
        },
        "EndTime": {
          "type": "number"
        }
      }
    },
    "BlockSize": {
      "type": "integer"
```

```
    },
    "VisualObjectCount": {
      "type": "integer"
    },
    "VisualObjectsData": {
      "type": "object",
      "properties": {
        "VisualObjectID": {
          "type": "string"
        },
        "SpatialAttitude": {
          "$ref": "https://schemas.mpai.community/OSD/V1.0/data/SpatialAttitude.json"
        }
      }
    }
  }
}
```

7.5.2 Semantics

Table 20 provides the semantics of Visual Scene Descriptors.

Table 20 – Visual Scene Geometry Semantics

<table>
<tr><th>Label</th><th>Size</th><th>Description</th></tr>
<tr><td>HEADER</td><td>9 Bytes</td><td></td></tr>
<tr><td>• Standard</td><td>7 Bytes</td><td>The string OSD-VSD</td></tr>
<tr><td>• Version</td><td>1 Byte</td><td>Major version</td></tr>
<tr><td>• Subversion</td><td>1 Byte</td><td>Minor</td></tr>
<tr><td>VSDID</td><td>16 Bytes</td><td>UUID Identifier of the total set of Visual Scene Descriptors (uuid).</td></tr>
<tr><td>Time</td><td>17 Bytes</td><td>Collects various data expressed with bits</td></tr>
<tr><td>• TimeType</td><td>0 bit</td><td>0=Relative: time starts at 0000/00/00T00:00
1=Absolute: time starts at 1970/01/01T00:00.</td></tr>
<tr><td>• Reserved</td><td>1-7 bits</td><td>reserved</td></tr>
<tr><td>• StartTime</td><td>8 Bytes</td><td>Start time of current Visual Scene Descriptors (in microseconds).</td></tr>
<tr><td>• EndTime</td><td>8 Bytes</td><td>End time of current Visual Scene Descriptors (in microseconds).</td></tr>
<tr><td>BlockSize</td><td>4 Bytes</td><td></td></tr>
<tr><td>VisualObjectCount</td><td>1 Byte</td><td>Number of Visual Objects in Visual Scene.</td></tr>
<tr><td>VisualObjectsData</td><td>N1 Bytes</td><td>Data associated to each Visual Object.</td></tr>
<tr><td>• VisualObjectID</td><td>1 Byte</td><td>ID of a specific Visual Object in a Visual Scene.</td></tr>
<tr><td>• Reserved</td><td>1 Byte</td><td></td></tr>
<tr><td>• SpatialAttitudeMask</td><td>2 Bytes</td><td>3*3 matrix of booleans (by rows)<table>
<tr><td></td><td>Position</td><td>Velocity</td><td>Acceleration</td></tr>
<tr><td>Cartesian</td><td></td><td></td><td></td></tr>
<tr><td>Spherical</td><td></td><td></td><td></td></tr>
<tr><td>Orientation</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>• SpatialAttitude</td><td>N2 Bytes</td><td>N2=N1-N3-3</td></tr>
</table>

7.6 Visual Scene Descriptors

7.6.1 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
```

```
  "title": "Visual Scene Descriptors",
  "type": "object",
  "properties": {
    "Header": {
      "type": "object",
      "properties": {
        "Standard": {
          "type": "string"
        },
        "Version": {
          "type": "integer"
        },
        "Subversion": {
          "type": "number"
        }
      }
    },
    "VSDID": {
      "type": "string"
    },
    "Time": {
      "type": "object",
      "properties": {
        "TimeType": {
          "type": "boolean"
        },
        "StartTime": {
          "type": "number"
        },
        "EndTime": {
          "type": "number"
        }
      }
    },
    "BlockSize": {
      "type": "integer"
    },
    "VisualObjectCount": {
      "type": "integer"
    },
    "VisualObjectsData": {
      "type": "object",
      "properties": {
        "VisualObjectID": {
          "type": "string"
        },
        "SpatialAttitude": {
          "$ref": "https://schemas.mpai.community/OSD/V1.0/data/SpatialAttitude.json"
        },
        "VisualObject": {
          "type": "object",
          "properties": {
            "FormatID": {
              "type": "integer"
            },
            "ObjectLength": {
              "type": "integer"
            },
            "DataInObject": {
              "$ref": "https://schemas.mpai.community/OSD/V1.0/data/VisualObject.json"
            }
          }
        }
      }
    }
  }
}
```

7.6.2 Semantics

Table 21 provides the semantics of Visual Scene Descriptors.

Table 21 – Visual Scene Descriptors Semantics

<table>
<tr><th>Label</th><th>Size</th><th>Description</th></tr>
<tr><td>HEADER</td><td>9 Bytes</td><td></td></tr>
<tr><td>• Standard</td><td>7 Bytes</td><td>The string OSD-VSD</td></tr>
<tr><td>• Version</td><td>1 Byte</td><td>Major version</td></tr>
<tr><td>• Subversion</td><td>1 Byte</td><td>Minor</td></tr>
<tr><td>VSDID</td><td>16 Bytes</td><td>UUID Identifier of the total set of Visual Scene Descriptors (uuid).</td></tr>
<tr><td>Time</td><td>17 Bytes</td><td>Collects various data expressed with bits</td></tr>
<tr><td>• TimeType</td><td>0 bit</td><td>0=Relative: time starts at 0000/00/00T00:00
1=Absolute: time starts at 1970/01/01T00:00.</td></tr>
<tr><td>• Reserved</td><td>1-7 bits</td><td>reserved</td></tr>
<tr><td>• StartTime</td><td>8 Bytes</td><td>Start time of current Visual Scene Descriptors (in microseconds).</td></tr>
<tr><td>• EndTime</td><td>8 Bytes</td><td>End time of current Visual Scene Descriptors (in microseconds).</td></tr>
<tr><td>BlockSize</td><td>4 Bytes</td><td></td></tr>
<tr><td>VisualObjectCount</td><td>1 Byte</td><td>Number of Visual Objects in Visual Scene.</td></tr>
<tr><td>VisualObjectsData</td><td>N1 Bytes</td><td>Data associated to each Visual Object.</td></tr>
<tr><td>• VisualObjectID</td><td>1 Byte</td><td>ID of a specific Visual Object in a Visual Scene.</td></tr>
<tr><td>• Reserved</td><td>1 Byte</td><td></td></tr>
<tr><td>• SpatialAttitudeMask</td><td>2 Bytes</td><td>3*3 matrix of booleans (by rows)
<table>
<tr><td></td><td>Position</td><td>Velocity</td><td>Acceleration</td></tr>
<tr><td>Cartesian</td><td></td><td></td><td></td></tr>
<tr><td>Spherical</td><td></td><td></td><td></td></tr>
<tr><td>Orientation</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>• SpatialAttitude</td><td>N2 Bytes</td><td>According to MPAI-OSD V1</td></tr>
<tr><td>• VisualObject</td><td>N3 Bytes</td><td></td></tr>
<tr><td>○ FormatID</td><td>1 Byte</td><td>Visual Object Format Identifier</td></tr>
<tr><td>○ Length</td><td>4 Bytes</td><td>Number of Bytes in Visual Object</td></tr>
<tr><td>○ DataInObject</td><td>N4 Bytes</td><td>Data of Visual Object</td></tr>
</table>

7.7 Audio-Visual Scene Geometry

7.7.1 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Audio-Visual Scene Geometry",
  "type": "object",
  "properties": {
    "Header": {
      "type": "object",
      "properties": {
        "Standard": {
          "type": "string"
        },
        "Version": {
          "type": "integer"
        },
        "Subversion": {
          "type": "integer"
        }
      }
    },
```

```
    "AVDID": {
      "type": "string"
    },
    "Time": {
      "type": "object",
      "properties": {
        "TimeType": {
          "type": "boolean"
        },
        "StartTime": {
          "type": "number"
        },
        "EndTime": {
          "type": "number"
        }
      }
    },
    "BlockSize": {
      "type": "integer"
    },
    "AVObjectCount": {
      "type": "integer"
    },
    "AVObjectsData": {
      "type": "object",
      "properties": {
        "AVObjectID": {
          "type": "string"
        },
        "SamplingRate": {
          "type": "number"
        },
        "SamplingType": {
          "type": "number"
        },
        "SpatialAttitude": {
          "$ref": "https://schemas.mpai.community/OSD/V1.0/data/SpatialAttitude.json"
        }
      }
    }
  }
}
```

7.7.2 Semantics

Table 22 provides the semantics of the Audio-Visual Scene Geometry.

Table 22 – Audio-Visual Scene Geometry

Label	Size	Description
HEADER	9 Bytes	
• Standard	7 Bytes	The string OSD-VSD
• Version	1 Byte	Major version
• Subversion	1 Byte	Minor
AVDID	16 Bytes	UUID Identifier of the total set of Audio-Visual Scene Descriptors.
Time	17 Bytes	Collects various data expressed with bits
• TimeType	0 bit	0=Relative: time starts at 0000/00/00T00:00 1=Absolute: time starts at 1970/01/01T00:00.
• Reserved	1-7 bits	reserved
• StartTime	8 Bytes	Start time of current Audio-Visual Scene Descriptors (in microseconds).
• EndTime	8 Bytes	End time of current Audio-Visual Scene Descriptors (in microseconds).

<table>
<tr><td>BlockSize</td><td>4 Bytes</td><td>Minimum BlockSize: ≥ 256 (uint32).</td></tr>
<tr><td>AVObjectCount</td><td>1 Byte</td><td>Number of Objects in Scene.</td></tr>
<tr><td>AVObjectData</td><td>N1 Bytes</td><td>Data associated to each Object.</td></tr>
<tr><td>• AVObjectID</td><td>1 Byte</td><td>ID of a specific Object in the Scene.</td></tr>
<tr><td>• SamplingRate</td><td>0-3 bits</td><td>0: 8kHz, 1: 16kHz, 2: 24kHz, 3: 32kHz, 4: 44.1kHz, 5: 48kHz, 6: 64kHz, 7: 96kHz, 8: 192kHz</td></tr>
<tr><td>• SampleType</td><td>4-5 bits</td><td>0:16bit, 1:24bit, 2:32bit, 3:64bit)</td></tr>
<tr><td>• Reserved</td><td>6-7 bits</td><td></td></tr>
<tr><td>• SpatialAttitudeMask</td><td>2 Bytes</td><td>3*3 matrix of booleans (by rows)
<table>
<tr><td></td><td>Position</td><td>Velocity</td><td>Acceleration</td></tr>
<tr><td>Cartesian</td><td></td><td></td><td></td></tr>
<tr><td>Spherical</td><td></td><td></td><td></td></tr>
<tr><td>Orientation</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>• SpatialAttitude</td><td>N2 Bytes</td><td>N2=N1-N3(or N4)-3</td></tr>
</table>

7.8 Audio-Visual Scene Descriptors

7.8.1 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Audio-Visual Scene Descriptors",
  "type": "object",
  "properties": {
    "Header": {
      "type": "object",
      "properties": {
        "Standard": {
          "type": "string"
        },
        "Version": {
          "type": "integer"
        },
        "Subversion": {
          "type": "integer"
        }
      }
    },
    "AVDID": {
      "type": "string"
    },
    "Time": {
      "type": "object",
      "properties": {
        "TimeType": {
          "type": "boolean"
        },
        "StartTime": {
          "type": "number"
        },
        "EndTime": {
          "type": "number"
        }
      }
    },
    "BlockSize": {
      "type": "integer"
    },
    "AVObjectCount": {
      "type": "integer"
    },
    "AVObjectsData": {
      "type": "object",
      "properties": {
```

```
        "AVObjectID": {
          "type": "string"
        },
        "SamplingRate": {
          "type": "number"
        },
        "SamplingType": {
          "type": "number"
        },
        "SpatialAttitude": {
          "$ref": "https://schemas.mpai.community/OSD/V1.0/data/SpatialAttitude.json"
        },
        "AVObject": {
          "type": "object",
          "properties": {
            "FormatID": {
              "type": "integer"
            },
            "ObjectLength": {
              "type": "integer"
            },
            "DataInAObject": {
              "$ref": "https://schemas.mpai.community/CAE/V2.1/data/AudioObject.json"
            },
            "DataInVObject": {
              "$ref": "https://schemas.mpai.community/OSD/V1.0/data/VisualObject.json"
            }
          }
        }
      }
    }
  }
}
```

7.8.2 Semantics

Table 23 provides the semantics of the Audio-Visual Scene Descriptors.

Table 23 – Audio-Visual Scene Descriptors

Label	Size	Description
HEADER	9 Bytes	
• Standard	7 Bytes	The string OSD-VSD
• Version	1 Byte	Major version
• Subversion	1 Byte	Minor
AVDID	16 Bytes	UUID Identifier of the total set of Audio-Visual Scene Descriptors.
Time	17 Bytes	Collects various data expressed with bits
• TimeType	0 bit	0=Relative: time starts at 0000/00/00T00:00 1=Absolute: time starts at 1970/01/01T00:00.
• Reserved	1-7 bits	reserved
• StartTime	8 Bytes	Start time of current Audio-Visual Scene Descriptors (in microseconds).
• EndTime	8 Bytes	End time of current Audio-Visual Scene Descriptors (in microseconds).
BlockSize	4 Bytes	Minimum BlockSize: ≥ 256 (uint32).
AVObjectCount	1 Byte	Number of Objects in Scene.
AVObjectData	N1 Bytes	Data associated to each Object.
• AVObjectID	1 Byte	ID of a specific Object in the Scene.
• SamplingRate	0-3 bits	0: 8kHz, 1: 16kHz, 2: 24kHz, 3: 32kHz, 4: 44.1kHz, 5: 48kHz, 6: 64kHz, 7: 96kHz, 8: 192kHz

<table>
<tr><td>• SampleType</td><td>4-5 bits</td><td>0:16bit, 1:24bit, 2:32bit, 3:64bit)</td></tr>
<tr><td>• Reserved</td><td>6-7 bits</td><td></td></tr>
<tr><td>• SpatialAttitudeMask</td><td>2 Bytes</td><td>3*3 matrix of booleans (by rows)<table>
<tr><td></td><td>Position</td><td>Velocity</td><td>Acceleration</td></tr>
<tr><td>Cartesian</td><td></td><td></td><td></td></tr>
<tr><td>Spherical</td><td></td><td></td><td></td></tr>
<tr><td>Orientation</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>• SpatialAttitude</td><td>N2 Bytes</td><td>According to MPAI-OSD V1</td></tr>
<tr><td>• AudioObject</td><td>N3 Bytes</td><td></td></tr>
<tr><td>○ FormatID</td><td>1 Byte</td><td>Audio Object Format Identifier</td></tr>
<tr><td>○ Length</td><td>4 Bytes</td><td>Number of Bytes in Audio Object</td></tr>
<tr><td>○ DataInObject</td><td>N4 Bytes</td><td>Data of Audio Object</td></tr>
<tr><td>• VisualObject</td><td>N5 Bytes</td><td></td></tr>
<tr><td>○ FormatID</td><td>1 Byte</td><td>Visual Object Format Identifier</td></tr>
<tr><td>○ Length</td><td>4 Bytes</td><td>Number of Bytes in Audio Object</td></tr>
<tr><td>○ DataInObject</td><td>N6 Bytes</td><td>Data of Visual Object</td></tr>
</table>

Annex 1 - MPAI Basics

1 General

In recent years, Artificial Intelligence (AI) and related technologies have been introduced in a broad range of applications affecting the life of millions of people and are expected to do so much more in the future. As digital media standards have positively influenced industry and billions of people, so AI-based data coding standards are expected to have a similar positive impact. In addition, some AI technologies may carry inherent risks, e.g., in terms of bias toward some classes of users making the need for standardisation more important and urgent than ever.

The above considerations have prompted the establishment of the international, unaffiliated, not-for-profit Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) organisation with the mission to develop *AI-enabled data coding standards* to enable the development of AI-based products, applications, and services.

As a rule, MPAI standards include four documents: Technical Specification, Reference Software Specifications, Conformance Testing Specifications, and Performance Assessment Specifications. The last – and new in standardisation – type of Specification includes standard operating procedures that enable users of MPAI Implementations to make informed decision about their applicability based on the notion of Performance, defined as a set of attributes characterising a reliable and trustworthy implementation.

2 Governance of the MPAI Ecosystem

Technical Specification: Governance of the MPAI Ecosystem lays down the foundations of the MPAI Ecosystem. MPAI develops and maintains the following documents the following technical documents :

1. Technical Specification.
2. Reference Software Specification.
3. Conformance Testing.
4. Performance Assessment.
5. Technical Report

An MPAI Standard is a collection of a variable number of the 5 document types.

Figure 21 depicts the operation of the MPAI ecosystem generated by MPAI Standards.

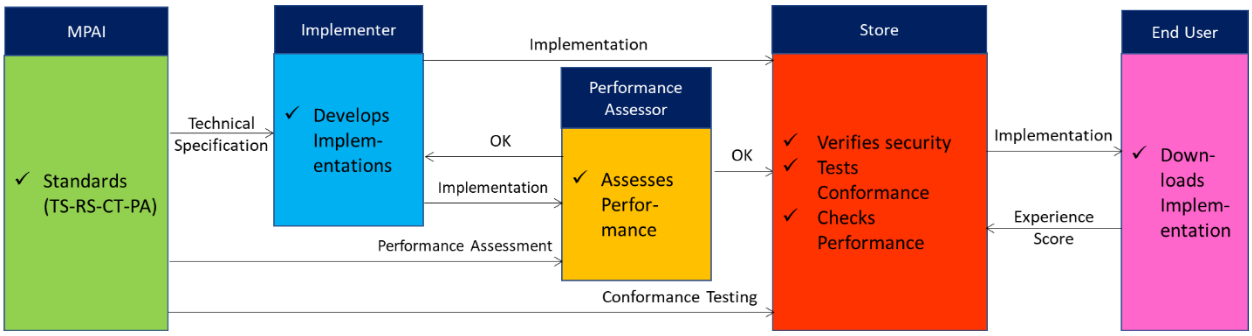

Figure 21 – The MPAI ecosystem operation

Table 24 identifies the following roles in the MPAI Ecosystem:

Table 24 - Roles in the MPAI Ecosystem

MPAI	Publishes Standards. Establishes the not-for-profit MPAI Store. Appoints Performance Assessors.
Implementers	Submit Implementations to Performance Assessors.
Performance Assessors	Inform Implementation submitters and the MPAI Store if Implementation Performance is acceptable.
Implementers	Submit Implementations to the MPAI Store.
MPAI Store	Assign unique ImplementerIDs (IID) to Implementers in its capacity as ImplementerID Registration Authority (IIDRA)[1]. Verifies security and Tests Implementation Conformance.
Users	Download Implementations and report their experience to MPAI.

3 AI Framework

In general, MPAI Application Standards are defined as aggregations – called AI Workflows (AIW) – of processing elements – called AI Modules (AIM) – executed in an AI Framework (AIF). MPAI defines Interoperability as the ability to replace an AIW or an AIM Implementation with a functionally equivalent Implementation.

Figure 22 depicts the MPAI-AIF Reference Model under which Implementations of MPAI Application Standards and user-defined MPAI-AIF Conforming applications operate [2].

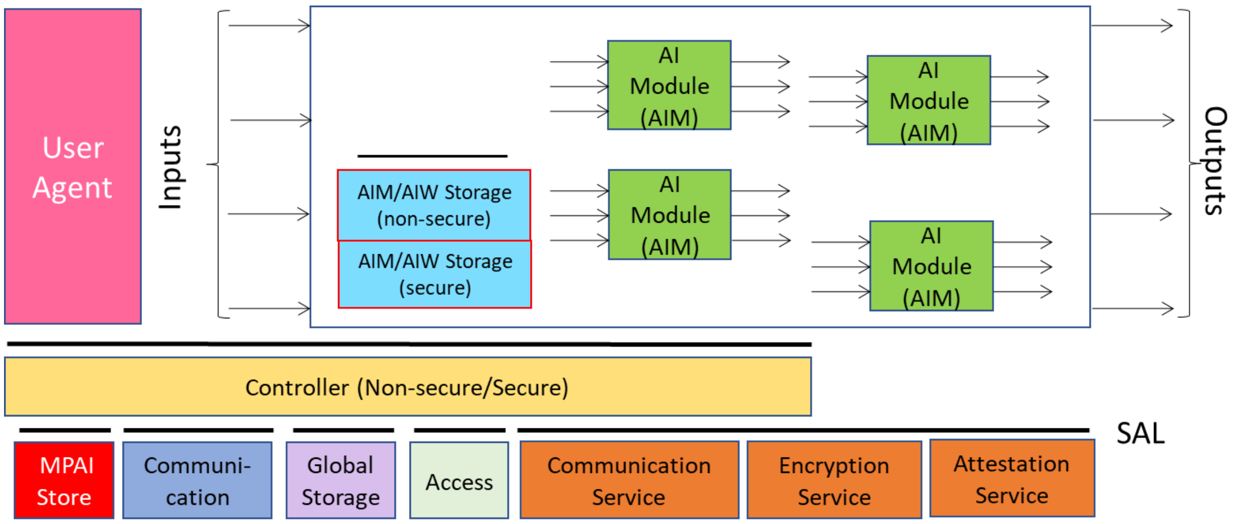

Figure 22 – The AI Framework (AIF) Reference Model

MPAI Application Standards normatively specify the Syntax and Semantics of the input and output data and the Function of the AIW and the AIMs, and the Connections between and among the AIMs of an AIW.

An AIW is defined by its Function and input/output Data and by its AIM topology. Likewise, an AIM is defined by its Function and input/output Data. MPAI standard are silent on the technology

[1] At the time of publication of this Technical Report, the MPAI Store was assigned as the IIDRA.

used to implement the AIM which may be based on AI or data processing, and implemented in software, hardware or hybrid software and hardware technologies.

MPAI also defines 3 Interoperability Levels of an AIF that executes an AIW. Table 25 gives the characteristics of an AIW and its AIMs of a given Level:

Table 25 - MPAI Interoperability Levels

Level	AIW	AIMs
1	An implementation of a use case	Implementations able to call the MPAI-AIF APIs.
2	An Implementation of an MPAI Use Case	Implementations of the MPAI Use Case
3	An Implementation of an MPAI Use Case certified by a Performance Assessor	Implementations of the MPAI Use Case certified by Performance Assessors

4 Audio Scene Description

The ability to describe (i.e., digitally represent) an audio-visual scene is a key requirement of several Use Cases. *Technical Specification: Context-based Audio Enhancement (MPAI-CAE) V2.1* [9] that includes the specification of Audio Scene Descriptors produced by the Composite Audio Scene Description AI Module (AIM) and depicted in Figure 28.

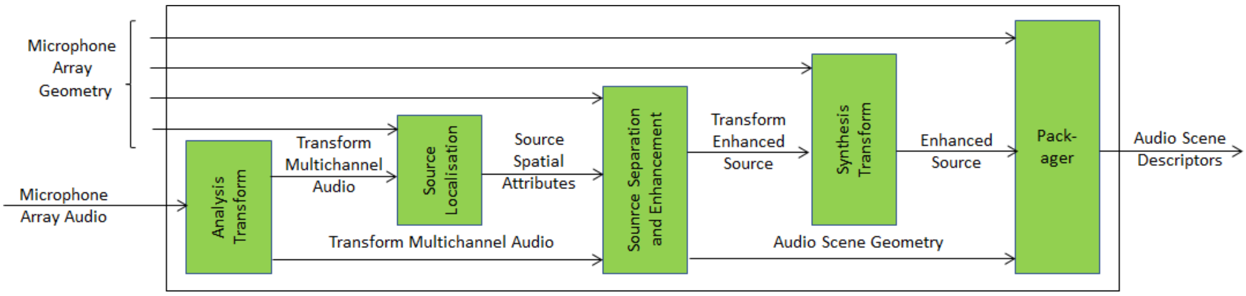

Figure 23 - The Audio Scene Description Composite AIM

5 Avatar-Based Videoconference

Technical Report: Avatar-Based Videoconference (MPAI-ARA) specifies AIWs and AIMs of a Use Case where geographically distributed humans hold a videoconference represented by their avatars having their visual appearance and uttering their real voice (Figure 24).

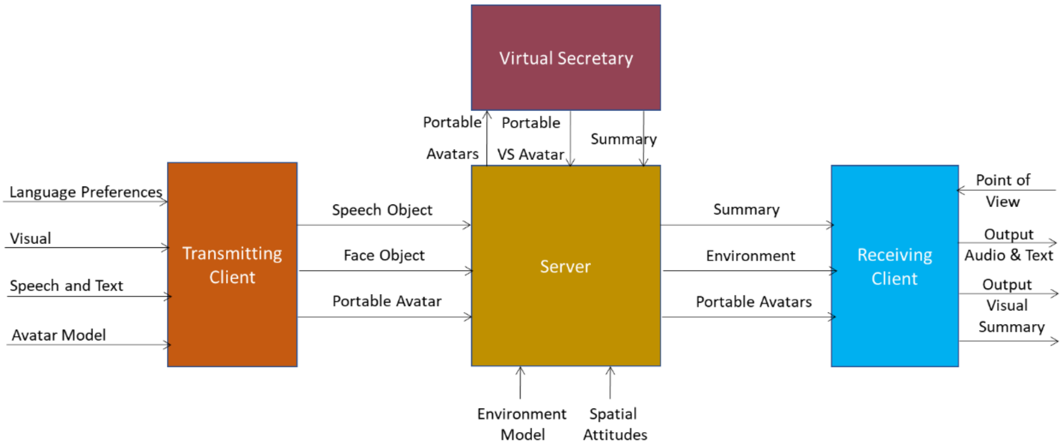

Figure 24 – Avatar-Based Videoconference end-to-end diagram

Figure 25 contains the reference architectures of the four AW Workflows constituting the Avatar-Based Videoconference: Client (Transmission side), Server, Virtual Secretary, and Client (Receiving side).

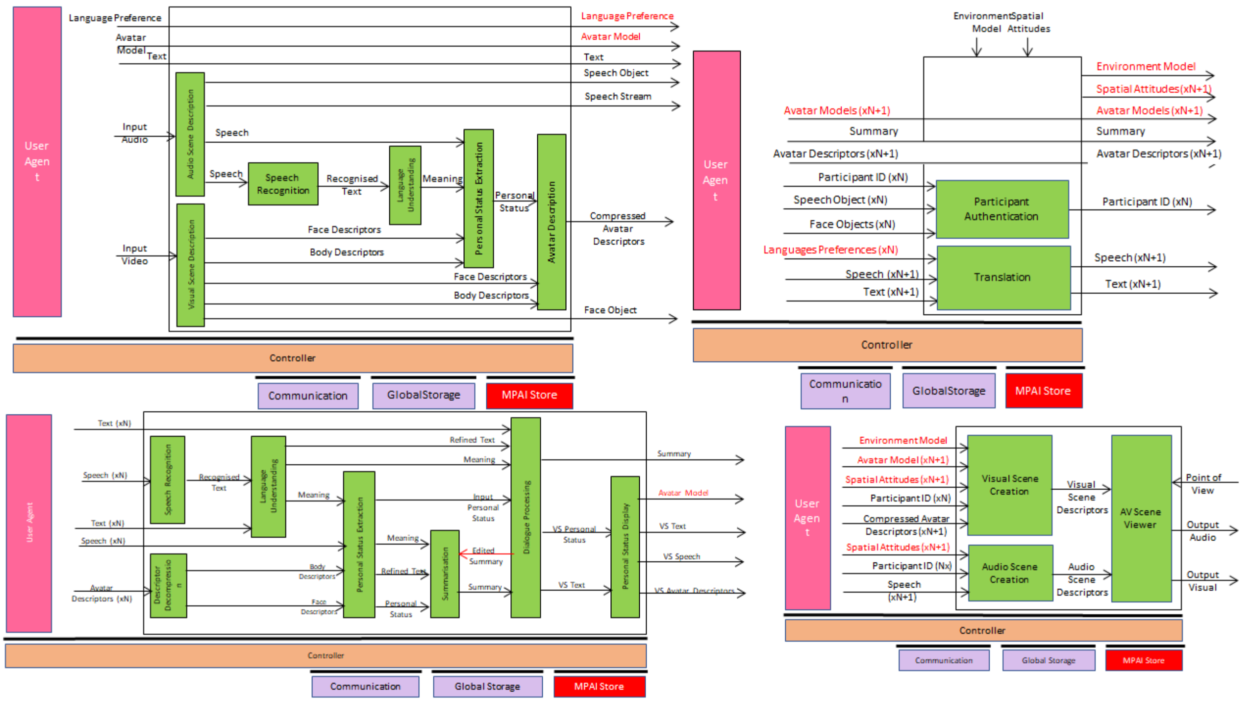

Figure 25 - The AIWs of Avatar-Based Videoconference

6 Connected Autonomous Vehicles

MPAI defines a Connected Autonomous Vehicle (CAV), as a physical system that:

1. Converses with humans by understanding their utterances, e.g., a request to be taken to a destination.
2. Acquires information with a variety of sensors on the physical environment where it is located or traverses like the one depicted in *Figure 26*.
3. Plans a Route enabling the CAV to reach the requested destination.
4. Autonomously reaches the destination by:

4.1. Moving in the physical environment.
4.2. Building Digital Representations of the Environment.
4.3. Exchanging elements of such Representations with other CAVs and CAV-aware entities.
4.4. Making decisions about how to execute the Route.
4.5. Acting on the CAV motion actuation to implement the decisions.

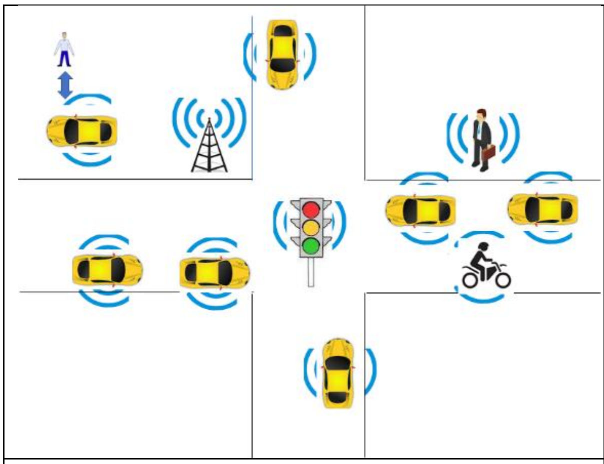

Figure 26 - An environment of CAV operation

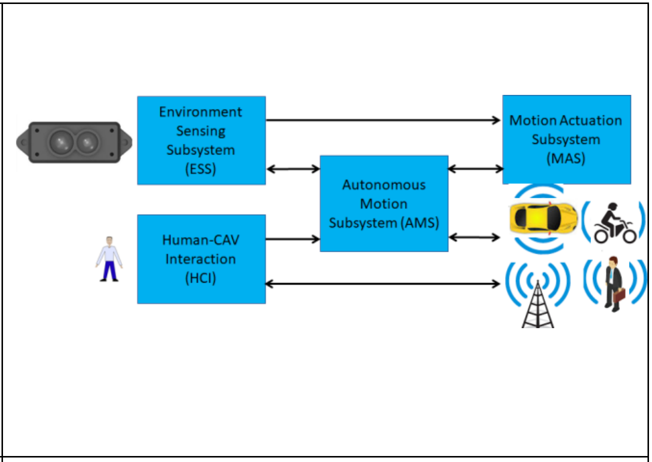

Figure 27 – The MPAI-CAV subsystems

MPAI believes in the capability of standards to accelerate the creation of a global competitive CAV market and has published Technical Specification: Connected Autonomous Vehicle (MPAI-CAV) – Architecture that includes (see *Figure 27*):

1. A CAV Reference Model broken down into four Subsystems.
2. The Functions of each Subsystem.
3. The Data exchanged between Subsystems.
4. A breakdown of each Subsystem in Components (see Figure 28) of which the following is specified:
   - 4.1. The Functions of the Components.
   - 4.2. The Data exchanged between Components.
   - 4.3. The Topology of Components and their Connections.
5. Functional Requirements of the Data exchanged (under development).
6. Standard technologies for the Data exchanged (in the future).

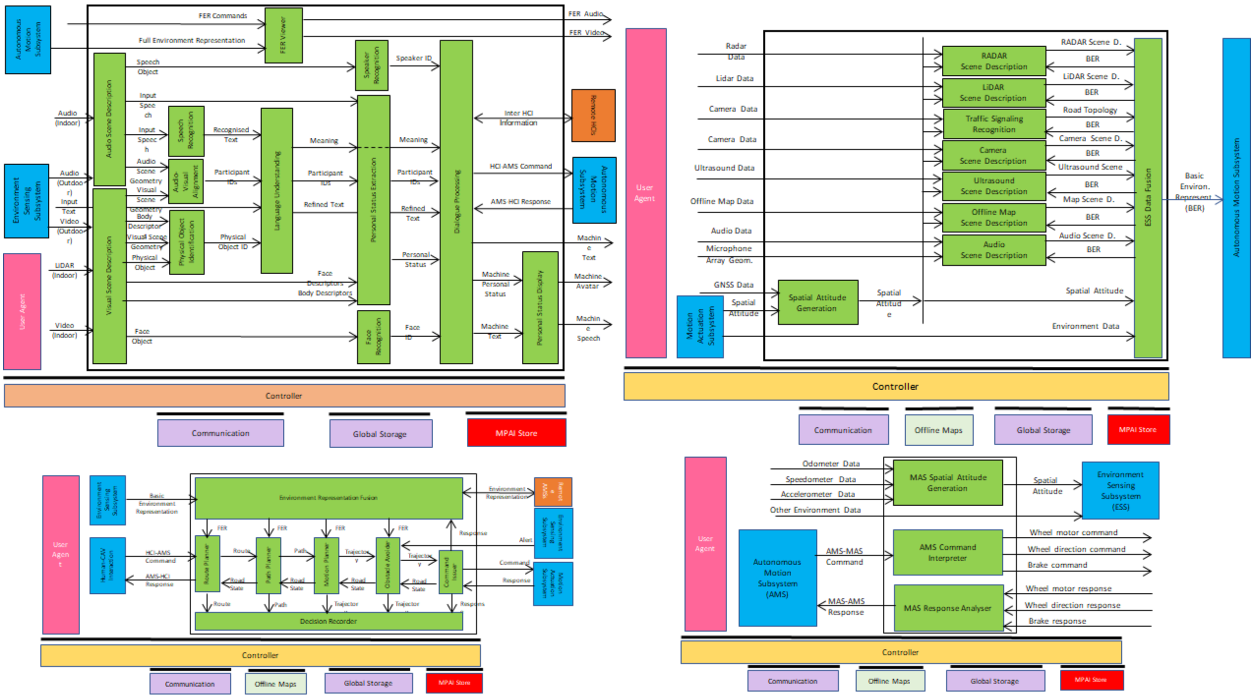

Figure 28 - The MPAI-CAV Subsystems with their Components (left-right & top bottom: Human-Cav Interaction, Environment Sensing Subsystem, Autonomous Motion Subsystem, and Motion Actuation Subsystem)

Subsystems are implemented as AI Workflows and Components as AI Modules according to Technical Specification: AI Framework (MPAI-AIF) [4].

Annex 2 - MPAI-wide terms and definitions

The Terms used in this standard whose first letter is capital and are not already included in Table 1 are defined in Table 26.

Table 26 - MPAI-wide Terms

Term	Definition
Access	Static or slowly changing data that are required by an application such as domain knowledge data, data models, etc.
AI Framework (AIF)	The environment where AIWs are executed.
AI Modules (AIM)	A data processing element receiving AIM-specific Inputs and producing AIM-specific Outputs according to according to its Function. An AIM may be an aggregation of AIMs.
AI Workflow (AIW)	A structured aggregation of AIMs implementing a Use Case receiving AIW-specific inputs and producing AIW-specific outputs according to the AIW Function.
Application Standard	An MPAI Standard designed to enable a particular application domain.
Channel	A connection between an output port of an AIM and an input port of an AIM. The term "connection" is also used as synonymous.
Communication	The infrastructure that implements message passing between AIMs
Composite AIM	An AIM aggregating more than one AIM.
Component	One of the 7 AIF elements: Access, Communication, Controller, Internal Storage, Global Storage, Store, and User Agent
Conformance	The attribute of an Implementation of being a correct technical Implementation of a Technical Specification.
Conformance Tester	An entity Testing the Conformance of an Implementation.
Conformance Testing	The normative document specifying the Means to Test the Conformance of an Implementation.
Conformance Testing Means	Procedures, tools, data sets and/or data set characteristics to Test the Conformance of an Implementation.
Connection	A channel connecting an output port of an AIM and an input port of an AIM.
Controller	A Component that manages and controls the AIMs in the AIF, so that they execute in the correct order and at the time when they are needed
Data Format	The standard digital representation of data.
Data Semantics	The meaning of data.
Ecosystem	The ensemble of actors making it possible for a User to execute an application composed of an AIF, one or more AIWs, each with one or more AIMs potentially sourced from independent implementers.
Explainability	The ability to trace the output of an Implementation back to the inputs that have produced it.
Fairness	The attribute of an Implementation whose extent of applicability can be assessed by making the training set and/or network open to testing for bias and unanticipated results.
Function	The operations effected by an AIW or an AIM on input data.

Global Storage	A Component to store data shared by AIMs.
Internal Storage	A Component to store data of the individual AIMs.
Identifier	A name that uniquely identifies an Implementation.
Implementation	1. An embodiment of the MPAI-AIF Technical Specification, or 2. An AIW or AIM of a particular Level (1-2-3) conforming with a Use Case of an MPAI Application Standard.
Implementer	A legal entity implementing MPAI Technical Specifications.
ImplementerID (IID)	A unique name assigned by the ImplementerID Registration Authority to an Implementer.
ImplementerID Registration Authority (IIDRA)	The entity appointed by MPAI to assign ImplementerID's to Implementers.
Interoperability	The ability to functionally replace an AIM with another AIW having the same Interoperability Level
Interoperability Level	The attribute of an AIW and its AIMs to be executable in an AIF Implementation and to: 1. Be proprietary (Level 1) 2. Pass the Conformance Testing (Level 2) of an Application Standard 3. Pass the Performance Testing (Level 3) of an Application Standard.
Knowledge Base	Structured and/or unstructured information made accessible to AIMs via MPAI-specified interfaces
Message	A sequence of Records transported by Communication through Channels.
Normativity	The set of attributes of a technology or a set of technologies specified by the applicable parts of an MPAI standard.
Performance	The attribute of an Implementation of being Reliable, Robust, Fair and Replicable.
Performance Assessment	The normative document specifying the Means to Assess the Grade of Performance of an Implementation.
Performance Assessment Means	Procedures, tools, data sets and/or data set characteristics to Assess the Performance of an Implementation.
Performance Assessor	An entity Assessing the Performance of an Implementation.
Profile	A particular subset of the technologies used in MPAI-AIF or an AIW of an Application Standard and, where applicable, the classes, other subsets, options and parameters relevant to that subset.
Record	A data structure with a specified structure
Reference Model	The AIMs and theirs Connections in an AIW.
Reference Software	A technically correct software implementation of a Technical Specification containing source code, or source and compiled code.
Reliability	The attribute of an Implementation that performs as specified by the Application Standard, profile and version the Implementation refers to, e.g., within the application scope, stated limitations, and for the period of time specified by the Implementer.
Replicability	The attribute of an Implementation whose Performance, as Assessed by a Performance Assessor, can be replicated, within an agreed level, by another Performance Assessor.
Robustness	The attribute of an Implementation that copes with data outside of the stated application scope with an estimated degree of confidence.

Scope	The domain of applicability of an MPAI Application Standard
Service Provider	An entrepreneur who offers an Implementation as a service (e.g., a recommendation service) to Users.
Standard	The ensemble of Technical Specification, Reference Software, Conformance Testing and Performance Assessment of an MPAI application Standard.
Technical Specification	(Framework) the normative specification of the AIF. (Application) the normative specification of the set of AIWs belonging to an application domain along with the AIMs required to Implement the AIWs that includes: 1. The formats of the Input/Output data of the AIWs implementing the AIWs. 2. The Connections of the AIMs of the AIW. 3. The formats of the Input/Output data of the AIMs belonging to the AIW.
Testing Laboratory	A laboratory accredited to Assess the Grade of Performance of Implementations.
Time Base	The protocol specifying how Components can access timing information
Topology	The set of AIM Connections of an AIW.
Use Case	A particular instance of the Application domain target of an Application Standard.
User	A user of an Implementation.
User Agent	The Component interfacing the user with an AIF through the Controller
Version	A revision or extension of a Standard or of one of its elements.

Annex 3 - Notices and Disclaimers Concerning MPAI Standards (Informative)

The notices and legal disclaimers given below shall be borne in mind when downloading and using approved MPAI Standards.

In the following, "Standard" means the collection of four MPAI-approved and published documents: "Technical Specification", "Reference Software" and "Conformance Testing" and, where applicable, "Performance Testing".

Life cycle of MPAI Standards

MPAI Standards are developed in accordance with the MPAI Statutes. An MPAI Standard may only be developed when a Framework Licence has been adopted. MPAI Standards are developed by especially established MPAI Development Committees who operate on the basis of consensus, as specified in Annex 1 of the MPAI Statutes. While the MPAI General Assembly and the Board of Directors administer the process of the said Annex 1, MPAI does not independently evaluate, test, or verify the accuracy of any of the information or the suitability of any of the technology choices made in its Standards.

MPAI Standards may be modified at any time by corrigenda or new editions. A new edition, however, may not necessarily replace an existing MPAI standard. Visit the web page to determine the status of any given published MPAI Standard.

Comments on MPAI Standards are welcome from any interested parties, whether MPAI members or not. Comments shall mandatorily include the name and the version of the MPAI Standard and, if applicable, the specific page or line the comment applies to. Comments should be sent to the MPAI Secretariat. Comments will be reviewed by the appropriate committee for their technical relevance. However, MPAI does not provide interpretation, consulting information, or advice on MPAI Standards. Interested parties are invited to join MPAI so that they can attend the relevant Development Committees.

Coverage and Applicability of MPAI Standards

MPAI makes no warranties or representations of any kind concerning its Standards, and expressly disclaims all warranties, expressed or implied, concerning any of its Standards, including but not limited to the warranties of merchantability, fitness for a particular purpose, non-infringement etc. MPAI Standards are supplied "AS IS".

The existence of an MPAI Standard does not imply that there are no other ways to produce and distribute products and services in the scope of the Standard. Technical progress may render the technologies included in the MPAI Standard obsolete by the time the Standard is used, especially in a field as dynamic as AI. Therefore, those looking for standards in the Data Compression by Artificial Intelligence area should carefully assess the suitability of MPAI Standards for their needs.

IN NO EVENT SHALL MPAI BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND

Annex 4 - Patent declarations (Informative)

Technical Specification: Object and Scene Description (MPAI-OSD) V1 has been developed according to the process outlined in the MPAI Statutes [9] and the MPAI Patent Policy [10], and following the prescriptions of *Framework Licence: Object and Scene Description (MPAI-OSD)* [11].

Table 27 will report the list of entities who will agree to licence their standard essential patents reading on *Technical Specification: Object and Scene Description (MPAI-OSD) V1* according to [11].:

Table 27 - Companies having submitted a patent declaration on (MPAI-OSD)

Entity	**Name**	**Email address**

The declarations2 will be published when patent declarations will be received in response to requests for declarations.

Annex 5 - JSON Metadata

1 Visual Spatial Object Identification (OSD-VOI)


```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-OSD",
      "AIW": "",
      "AIM": "VisualSpatialObjectIdentification",
      "Version": "1"
    },
    "Description": "This AIM provides the ID of the Instance of the Visual Spatial Object indi-
cated by a human or avatar.",
    "Types": [
      {
        "Name": "BodyDescriptors_t",
        "Type": "uint8[]}"
      },
      {
        "Name": "VisualSceneGeometry_t",
        "Type": "uint8[]}"
      },
      {
        "Name": "VisualObject_t",
        "Type": "uint8[]"
      },
      {
        "Name": "InstanceID_t",
        "Type": "Audio_t[]"
      }
    ],
    "Ports": [
      {
        "Name": "BodyDescriptors",
        "Direction": "InputOutput",
        "RecordType": " BodyDescriptors_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualSceneDescriptors",
        "Direction": "InputOutput",
        "RecordType": "VisualSceneDescriptors_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualObject",
        "Direction": "InputOutput",
        "RecordType": "VisualObject_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualInstanceID",
        "Direction": "OutputInput",
        "RecordType": "VisualInstanceID_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      }
    ],
    "SubAIMs": [
      {
        "Name": "VisualDirectionIdentification",
```

```
      "Identifier": {
        "ImplementerID": "/* String assigned by IIDRA */",
        "Specification": {
          "Standard": "MPAI-CAE",
          "AIW": "",
          "AIM": "VisualDirectionIdentification",
          "Version": "1"
        }
      }
    },
    {
      "Name": "VisualObjectExtraction",
      "Identifier": {
        "ImplementerID": "/* String assigned by IIDRA */",
        "Specification": {
          "Standard": "MPAI-OSD",
          "AIW": "",
          "AIM": "VisualObjectExtraction",
          "Version": "1"
        }
      }
    },
    {
      "Name": "ObjectInstanceIdentification",
      "Identifier": {
        "ImplementerID": "/* String assigned by IIDRA */",
        "Specification": {
          "Standard": "MPAI-MMC",
          "AIW": "",
          "AIM": "ObjectInstanceIdentification",
          "Version": "1"
        }
      }
    },
  ],
  "Topology": [
    {
      "_comment": "Input to first AIM column" },
    {
      "Output": {
        "AIMName": "",
        "PortName": "BodyDescriptors"
      },
      "Input": {
        "AIMName": "VisualDirectionIdentification",
        "PortName": "BodyDescriptors"
      }
    },
    {
      "_comment": "Input to second AIM column" },
    {
    {
      "Output": {
        "AIMName": "VisualDirectionIdentification",
        "PortName": "VisualObjectDirection"
      },
      "Input": {
        "AIMName": "Visual Object Extraction",
        "PortName": "VisualObjectDirection"
      }
    },
    {
      "_comment": "Input to third AIM column" },
    {
    {
      "Output": {
        "AIMName": "VisualObjectExtraction",
        "PortName": "TargetVisualObject"
      },
      "Input": {
        "AIMName": "Visual Object Extraction",
        "PortName": "TargetVisualObject"
      }
    },
```

```
      {
        "_comment": "Input to output" },
      {
      {
        "Output": {
          "AIMName": "ObjectInstanceIdentification",
          "PortName": "VisualInstanceID"
        },
        "Input": {
          "AIMName": "",
          "PortName": "VisualInstanceID"
        }
      },
    "Implementations": [],
    "Documentation": [
      {
        "Type": "Tutorial",
        "URI": "https://mpai.community/standards/mpai-osd/"
      }
    ]
  }
}
```


1.1 Visual Direction Identification


```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-OSD",
      "AIW": "",
      "AIM": "VisualDirectionIdentification",
      "Version": "1"
    },
    "Description": "This AIM identifies the Direction of the Visual Object crossed by the line traversing the finger used by the human or avatar to indicate the Visual Object.",
    "Types": [
      {
        "Name": "BodyDescriptors_t",
        "Type": "uint8[]"
      },
      {
         "Name":"VisualSceneGeometry_t",
         "Type":"{uint8[]"
      },
      {
        "Name": "VisualObjectDirection_t",
        "Type": "uint8[]"
      }
    ],
    "Ports": [
      {
        "Name": "BodyDescriptors",
        "Direction": "InputOutput",
        "RecordType": "BodyDescriptors_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualSceneGeometry",
        "Direction": "OutputInput",
        "RecordType": "VisualSceneGeometry_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualObjectDirection",
        "Direction": "OutputInput",
        "RecordType": "VisualObjectDirection_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
```

```
      }
    ],
    "SubAIMs": [],
    "Topology": [],
    "Implementations": [],
    "Documentation": [
      {
        "Type": "Tutorial",
      }
    ]
  }
}
```

1.2 Visual Object Extraction

```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-OSD",
      "AIW": "",
      "AIM": "VisualObjectExtraction",
      "Version": "1"
    },
    "Description": "This AIM identifies the Direction of the Visual Object crossed by the line
traversing the finger used by the human or avatar to indicate the Visual Object.",
    "Types": [
      {
        "Name": "VisualObjectDirection_t",
        "Type": "uint8[]"
      },
      {
         "Name":"VisualSceneGeometry_t",
         "Type":"{uint8[]"
      },
      {
        "Name": "VisualObject_t",
        "Type": "uint8[]"
      }
    ],
    "Ports": [
      {
        "Name": "VisualObjectDirection",
        "Direction": "InputOutput",
        "RecordType": "VisualObjectDirection_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualSceneGeometry",
        "Direction": "OutputInput",
        "RecordType": "VisualSceneGeometry_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "TargetVisualObject",
        "Direction": "OutputInput",
        "RecordType": "TargetVisualObject_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      }
    ],
    "SubAIMs": [],
    "Topology": [],
    "Implementations": [],
    "Documentation": [
      {
        "Type": "Tutorial",
      }
    ]
```

```
  }
}
```

1.3 Object Instance Identification

```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-OSD",
      "AIW": "",
      "AIM": "ObjectInstanceIdentification",
      "Version": "1"
    },
    "Description": "This AIM identifies the Visual Object instance.",
    "Types": [
      {
        "Name": "VisualObject_t",
        "Type": "uint8[]"
      },
      {
        "Name": "VisualInstanceID_t",
        "Type": "uint8[]"
      }
    ],
    "Ports": [
      {
        "Name": "TargetVisualObject",
        "Direction": "InputOutput",
        "RecordType": "TargetVisualObject_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualInstanceID",
        "Direction": "OutputInput",
        "RecordType": "VisualInstanceID_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      }
    ],
    "SubAIMs": [],
    "Topology": [],
    "Implementations": [],
    "Documentation": [
      {
        "Type": "Tutorial",
      }
    ]
  }
}
```

2 Audio-Visual Scene Description (OSD-AVD)

```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-OSD",
      "AIW": "",
      "AIM": "AVSceneDescription",
      "Version": "1"
    },
    "Description": "This AIM receives two independently developed Audio Scene Descriptors and
Visual Scene Descriptors in the same Virtual Space and produces Audio-Visual Scene Descriptors
whose co-located Audio Objects and Visual Objects have the same or related identifiers.",
    "Types": [
      {
        "Name": "Audio_t",
        "Type": "uint16[]"
      },
      {
```

```
      "Name": "ArrayAudio_t",
      "Type": "Audio_t"
    },
    {
       "Name":"Video_t",
       "Type":"{uint8[] Red; uint8[] Green; uint8[] Blue; uint8[]; uint16[] Depth}"
    },
    {
      "Name": "AVSceneDescriptors_t",
      "Type": "uint8[]}"
    }
  ],
  "Ports": [
    {
      "Name": "InputAudio",
      "Direction": "InputOutput",
      "RecordType": "ArrayAudio_t",
      "Technology": "Software",
      "Protocol": "",
      "IsRemote": false
    },
    {
      "Name": "InputVisual",
      "Direction": "InputOutput",
      "RecordType": "Visual_t",
      "Technology": "Software",
      "Protocol": "",
      "IsRemote": false
    },
    {
      "Name": "AVSceneDescriptors",
      "Direction": "InputOutput",
      "RecordType": "AVSceneDescriptors_t",
      "Technology": "Software",
      "Protocol": "",
      "IsRemote": false
    }
  ],
  "SubAIMs": [
    {
      "Name": "AudioSceneDescription",
      "Identifier": {
        "ImplementerID": "/* String assigned by IIDRA */",
        "Specification": {
          "Standard": "MPAI-CAE",
          "AIW": "",
          "AIM": "AudioSceneDescription",
          "Version": "2.1"
        }
      }
    },
    {
      "Name": "VisualSceneDescription",
      "Identifier": {
        "ImplementerID": "/* String assigned by IIDRA */",
        "Specification": {
          "Standard": "MPAI-OSD",
          "AIW": "",
          "AIM": "VisualSceneDescription",
          "Version": "1"
        }
      }
    },
    {
      "Name": "AVAlignment",
      "Identifier": {
        "ImplementerID": "/* String assigned by IIDRA */",
        "Specification": {
          "Standard": "MPAI-OSD",
          "AIW": "",
          "AIM": "AVAlignment",
          "Version": "1"
        }
      }
```

```
    },
    {
      "Name": "AVSceneMultiplexing",
      "Identifier": {
        "ImplementerID": "/* String assigned by IIDRA */",
        "Specification": {
          "Standard": "MPAI-OSD",
          "AIW": "",
          "AIM": "AVSceneMultiplexing",
          "Version": "1"
        }
      }
    }
  ],
  "Topology": [
    {
      "_comment": "Input to first AIM column"
    },
      "Output": {
        "AIMName": "",
        "PortName": "InputAudio"
      },
      "Input": {
        "AIMName": "AudioSceneDescription",
        "PortName": "InputAudio"
      }
    },
      "Output": {
        "AIMName": "",
        "PortName": "InputVisual"
      },
      "Input": {
        "AIMName": "VisualSceneDescription",
        "PortName": "InputVisual"
      }
    },
    {
      "_comment": "Input to second AIM column"
    },
    {
      "Output": {
        "AIMName": "AudioSceneDescription",
        "PortName": "AudioSceneGeometry"
      },
      "Input": {
        "AIMName": "AVAlignment",
        "PortName": "AudioSceneGeometry"
      }
    },
    {
      "Output": {
        "AIMName": "VisualSceneDescription",
        "PortName": "VisualSceneGeometry"
      },
      "Input": {
        "AIMName": "AVAlignment",
        "PortName": "VisualSceneGeometry"
      }
    },
    {
      "_comment": "Input to third AIM column"
    },
    {
      "Output": {
        "AIMName": "AudioSceneDescription",
        "PortName": "AudioObjects"
      },
      "Input": {
        "AIMName": "AVSceneMultiplexing",
        "PortName": " AudioObjects "
      }
    },
    {
      "Output": {
```

```
          "AIMName": "AVAlignment",
          "PortName": "AVSceneGeometry"
        },
        "Input": {
          "AIMName": "AVSceneMultiplexing",
          "PortName": "AVSceneGeometry"
        }
      },
      {
        "_comment": "Input to output"
      },
      {
        "Output": {
          "AIMName": "AVSceneMultiplexing",
          "PortName": "AVSceneDescription"
        },
        "Input": {
          "AIMName": "",
          "PortName": "AVSceneDescription"
        }
      }
    ],
    "Implementations": [],
    "Documentation": [
      {
        "Type": "Tutorial",
        "URI": "https://mpai.community/standards/mpai-osd/"
      }
    ]
  }
}
```

2.1 Audio Scene Description

https://schemas.mpai.community/CAE/V2.1/ASD/AudioSceneDescription.json

2.1.1 Audio Analysis Transform

https://schemas.mpai.community/CAE/V2.1/ASD/AAT/AudioAnalysisTransform.json

2.1.2 Audio Source Localisation

https://schemas.mpai.community/CAE/V2.1/ASD/ASL/AudioSourceLocalisation.json

2.1.3 Audio Separation and Enhancement

https://schemas.mpai.community/CAE/V2.1/ASD/ASE/AudioSeparationAndEnhancement.json

2.1.4 Audio Synthesis Transform

https://schemas.mpai.community/CAE/V2.1/ASD/AST/AudioSynthesisTransform.json

2.1.5 Audio Description Multiplexing

https://schemas.mpai.community/CAE/V2.1/ASD/ADM/AudioDescriptionMultiplexing.json

2.2 Visual Scene Description

```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-OSD",
      "AIW": "",
      "AIM": "VisualSceneDescription",
      "Version": "1"
    },
    "Description": "This AIM describes the Visual Objects in a Scene.",
    "Types": [
      {
         "Name":"Video_t",
         "Type":"{uint8[] Red; uint8[] Green; uint8[] Blue; uint8[]; uint16[] Depth}"
      },
```

```
        {
          "Name": "VisualSceneGeometry_t",
          "Type": "uint8[]"
        },
        {
          "Name": "VisualObject_t",
          "Type": "uint8[]"
        }
      ],
      "Ports": [
        {
          "Name": "InputVisual",
          "Direction": "InputOutput",
          "RecordType": "InputVisual_t",
          "Technology": "Software",
          "Protocol": "",
          "IsRemote": false
        },
        {
          "Name": "VisualSceneGeometry",
          "Direction": "OutputInput",
          "RecordType": "VisualSceneGeometry_t",
          "Technology": "Software",
          "Protocol": "",
          "IsRemote": false
        },
        {
          "Name": "VisualObject",
          "Direction": "InputOutput",
          "RecordType": "VisualObject_t",
          "Technology": "Software",
          "Protocol": "",
          "IsRemote": false
        }
      ],
      "SubAIMs": [],
      "Topology": [],
      "Implementations": [],
      "Documentation": [
        {
          "Type": "Tutorial",
        }
      ]
    }
}
```

2.3 Audio-Visual Alignment

```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-CAE",
      "AIW": "",
      "AIM": "AudioVisualAlignment",
      "Version": "2.1"
    },
    "Description": "This AIM identifies the Visual Object instance.",
    "Types": [
      {
        "Name": "AudioSceneGeometry_t",
        "Type": "uint8[]"
      },
      {
        "Name": "VisualSceneGeometry_t",
        "Type": "uint8[]"
      },
      {
        "Name": "AVSceneGeometry_t",
        "Type": "uint8[]"
      }
    ],
    "Ports": [
      {
```

```
        "Name": "InputVisual",
        "Direction": "InputOutput",
        "RecordType": "InputVisual_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualSceneGeometry",
        "Direction": "OutputInput",
        "RecordType": "AVSceneGeometry_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualObject",
        "Direction": "OutputInput",
        "RecordType": "VisualObject_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      }
    ],
    "SubAIMs": [],
    "Topology": [],
    "Implementations": [],
    "Documentation": [
      {
        "Type": "Tutorial",
        "URI": "https://mpai.community/standards/mpai-osd/"
      }
    ]
  }
}
```

2.4 AV Scene Multiplexing

```
{
  "Identifier": {
    "ImplementerID": "/* String assigned by IIDRA */",
    "Specification": {
      "Name": "MPAI-OSD",
      "AIW": "",
      "AIM": "AVSceneMultiplexing",
      "Version": "1"
    },
    "Description": "This AIM multiplexes the components of the AV Scene Descriptors.",
    "Types": [
      {
        "Name": "AudioObject_t",
        "Type": "uint8[]"
      },
      {
        "Name": "AVSceneGeometry_t",
        "Type": "uint8[]"
      },
      {
        "Name": "VisualObject_t",
        "Type": "uint8[]"
      },
      {
        "Name": "AVSceneDescriptors_t",
        "Type": "uint8[]"
      }
    ],
    "Ports": [
      {
        "Name": "AudioObject",
        "Direction": "InputOutput",
        "RecordType": "AudioObject_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
```

```
      },
      {
        "Name": "AVSceneGeometry",
        "Direction": "OutputInput",
        "RecordType": "AVSceneGeometry_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "VisualObject",
        "Direction": "InputOutput",
        "RecordType": "VisualObject_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      },
      {
        "Name": "AVSceneDescriptors",
        "Direction": "OutputInput",
        "RecordType": "AVSceneDescriptors_t",
        "Technology": "Software",
        "Protocol": "",
        "IsRemote": false
      }
    ],
    "SubAIMs": [],
    "Topology": [],
    "Implementations": [],
    "Documentation": [
      {
        "Type": "Tutorial",
        "URI": "https://mpai.community/standards/mpai-osd/"
      }
    ]
  }
}
```