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

MPAI Technical Specification

Connected Autonomous Vehicle (MPAI-CAV) Technologies (CAV-TEC)

V2.0

WARNING

Use of the technologies described in this Technical Specification 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 Specification.

Readers are invited to review Notices and Disclaimers.

Technical Specification
Connected Autonomous Vehicle (MPAI-CAV)
Technologies (CAV-TEC) V2.0

1 Foreword

The international, unaffiliated, non-profit ***Moving Picture, Audio, and Data Coding by Artificial Intelligence (MPAI)*** organisation was established in September 2020 in the context of:

1. **Increasing** use of Artificial Intelligence (AI) technologies applied to a broad range of domains affecting millions of people
2. **Marginal** reliance on standards in the development of those AI applications
3. **Unprecedented** impact exerted by standards on the digital media industry affecting billions of people

believing that AI-based data coding standards will have a similar positive impact on the Information and Communication Technology industry.

The design principles of the MPAI organisation as established by the MPAI Statutes are the development of AI-based Data Coding standards in pursuit of the following policies:

1. Publish upfront clear Intellectual Property Rights licensing frameworks.
2. Adhere to a rigorous standard development process.
3. Be friendly to the AI context but, to the extent possible, remain agnostic to the technology thus allowing developers freedom in the selection of the more appropriate – AI or Data Processing – technologies for their needs.
4. Be attractive to different industries, end users, and regulators.
5. Address five standardisation areas:
    1. *Data Type*, a particular type of Data, e.g., Audio, Visual, Object, Scenes, and Descriptors with as clear semantics as possible.
    2. *Qualifier*, specialised Metadata conveying information on Sub-Types, Formats, and Attributes of a Data Type.
    3. *AI Module* (AIM), processing elements with identified functions and input/output Data Types.

    4. *AI Workflow* (AIW), MPAI-specified configurations of AIMs with identified functions and input/output Data Types.
    5. *AI Framework* (AIF), an environment enabling dynamic configuration, initialisation, execution, and control of AIWs.
6. Provide appropriate Governance of the ecosystem created by MPAI Technical Specifications enabling users to:
    1. *Operate* Reference Software Implementations of MPAI Technical Specifications provided together with Reference Software Specifications
    2. *Test* the conformance of an implementation with a Technical Specification using the Conformance Testing Specification.
    3. *Assess* the performance of an implementation of a Technical Specification using the Performance Assessment Specification.
    4. *Obtain* conforming implementations possibly with a performance assessment report from a trusted source through the MPAI Store.

MPAI operates on four solid pillars:

1. The MPAI Patent Policy specifies the MPAI standard development process and the Framework Licence development guidelines.
2. *Technical Specification: Artificial Intelligence Framework (MPAI-AIF)* V2.1 specifies an environment enabling initialisation, dynamic configuration, and control of AIWs in the standard AI Framework environment depicted in Figure 1. An AI Framework can execute AI applications called AI Workflows (AIW) typically including interconnected AI Modules (AIM). MPAI-AIF supports small- and large-scale high-performance components and promotes solutions with improved explainability.

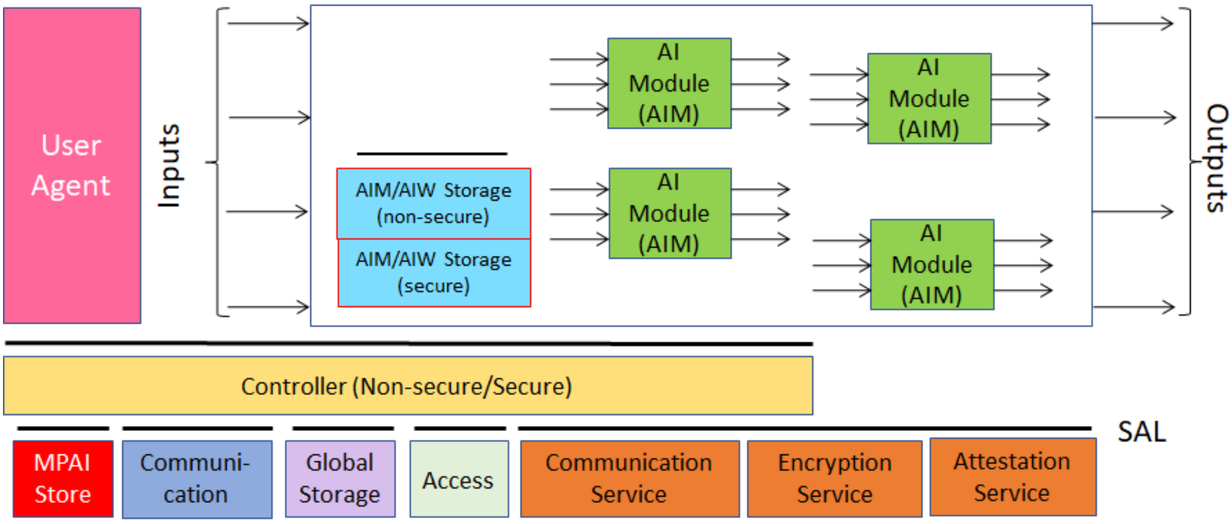

Figure 1 – The AI Framework (MPAI-AIF) V2 Reference Model

3. ***Technical Specification: Data Types, Formats, and Attributes (MPAI-TFA) V1.2*** specifies Qualifiers, a type of metadata supporting the operation of AIMs receiving data from other AIMs or from input data. Qualifiers convey information on Sub-Types (e.g., the type of colour), Formats (e.g., the type of compression and transport), and Attributes (e.g., semantic information in the Content). Although Qualifiers are human-readable, they are only intended to be used by AIMs. Therefore, Text, Speech, Audio, Visual, and other Data received by or exchanged between AIWs and AIMs should be interpreted as being composed of Content (Text, Speech, Audio, and Visual as appropriate) and associated Qualifiers. For instance, a Text Object is composed of Text Data and Text Qualifier. The specification of most MPAI Data Types reflects this point.

4. [*Technical Specification: Governance of the MPAI Ecosystem (MPAI-GME) V1.1*]() defines the following elements:
    1. Standards, i.e., the ensemble of Technical Specifications, Reference Software, Conformance Testing, and Performance Assessment.
    2. Developers of MPAI-specified AIMs and Integrators of MPAI-specified AIWS (Implementers).
    3. MPAI Store in charge of making AIMs and AIWs submitted by Implementers available to Integrators and End Users.
    4. Performance Assessors, independent entities assessing the performance of implementations in terms of Reliability, Replicability, Robustness, and Fairness.
    5. End Users.

The interaction between and among actors of the MPAI Ecosystem are depicted in Figure 2.

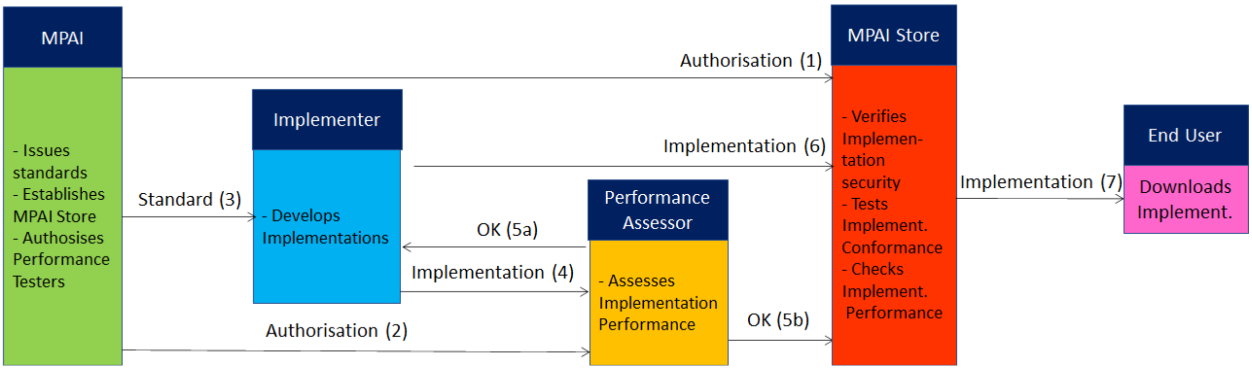

Figure 2 – The MPAI Ecosystem

2 Introduction

(Informative)

Since the invention of the first Motorwagen in 1885, many innovations have made automobiles easier to drive and more responsive to human needs. A short list includes electric ignition starter, car radio, car key, power steering, cruise control, electric windows, intermittent windshield wipers, automatic transmission, anti-lock braking system (ABS), digital dashboard displays, electromagnetic parking sensors, on-board diagnostics, mobile connection, satellite navigation, reversing camera, automatic parking, driver assistance features, etc.

Since the the first “self-driving” car attempt in 1939, many efforts have transformed automobiles from machines entirely driven by humans to machines with some “self-driving” capabilities. The "Levels" of the Society of Automotive Engineers (SAE) in the USA classify cars with some “self-driving” capabilities. Today, self-driving cars are not only technically possible, but commercially available. They promise to bring benefits that will positively affect industry, society, and the environment, such as:

1. *Saving lives* and *reducing injuries* by removing human error thanks to a machine less prone to errors.
2. *Giving* humans more time for rewarding activities, such as interpersonal communication.
3. *Optimising* the use of vehicles and infrastructure.
4. *Reducing* congestion and pollution.
5. *Supporting* elderly and disabled people.

Therefore, society and individuals will be positively impacted by the transformation of today's “niche market” into tomorrow's vibrant “mass market” of Connected Autonomous Vehicles.

MPAI believes that a market of standard-enabled interchangeable components can offer

affordable and safe Connected Autonomous Vehicles sooner and more efficiently than waiting for market forces to produce "monolithic" cars with progressively higher SAE Levels.
The MPAI-proposed open process is based on a shared *Reference Model* that partitions a CAV into *subsystems* and *components* specified in terms of functional requirements and exchanging data of known semantics. The Reference Model will help:

1. Researchers to optimise component technologies.
2. Component manufacturers to bring their standard-conforming components to market once they are mature.
3. Car manufacturers to access an open global market of interchangeable components.
4. Regulators to oversee conformance testing of components following standard procedures.
5. Users to rely on Connected Autonomous Vehicles whose operation they can explain to a large extent.

Far from being an impediment to technological progress, an interface-standard enables the creation of a competitive market offering increasingly performing components until a new, more powerful reference model will eventually replace the model with another, initiating a new sequence of performance improvements.
In this Introduction and in the following Chapters, Capitalised Terms are defined in Table 1 if they are specific to this Technical Specification or online if they are shared with other MPAI Technical Specifications.
Chapters and Sections are Normative unless they are labelled as Informative.

3 Scope

Technical Specification: Connected Autonomous Vehicle (MPAI-CAV) - Technologies (CAV-TEC) V1.0 - in the following called **CAV-TEC V1.0** or simply **CAV-TEC**: specifies the CAV-TEC Reference models and the following elements of a Connected Autonomous Vehicle:

CAV Subsystems (AIWs)	CAV Components (AIMs)	CAV Data Types
Functions	Functions	Definitions
Reference Model	Reference Model	Functional Requirements
I/O Data	I/O Data	Syntax
Functions of Components	Sub-components, if any	Semantics.
I/O Data of Components	JSON metadata	

to achieve the goals specified in the Introduction.

The CAV-TEC Technical Specification:

1. Reuses several AI Workflows and AI Modules specified by other MPAI Technical Specifications.
2. Assumes that Subsystems and Components are implemented as AI Workflows and AI Modules executed in the AI Framework specified by Technical Specification: AI Framework V2.1.
3. Does not specify the AIM internals nor the technologies used but only the functions and interfaces.
4. Does not mandate that a CAV implement CAV-TEC V1.0's specified AIMs; any AIM aggregation shall preserve external interfaces.
5. Is agnostic as to where - onboard or remotely - the specified processing functions are performed.

CAV-TEC has been developed by the Requirements (CAV) group of the Requirements Standing Committee. MPAI may develop new CAV-TEC Versions or new Technical Specifications whose scope falls within the CAV-TEC V1.0 Technical Specification.

4 Definitions

Table 1 defines the CAV-specific Terms used by CAV-TEC. All MPAI-defined Terms - some of which are used by CAV-TEC - are available online.

Table 1 - Terms and Definitions

Note A dash "-" preceding a Term in this Table means the following:

1. If the font is normal, the Term in the table without a dash and preceding the one with a dash should be placed before that Term. The notation is used to concentrate in one place all the Terms that are composed of, e.g., the word Audio followed by one of the words Object, Scene, and Scene Descriptors.
2. If the font is *italic*, the Term in the table without a dash and preceding the one with a dash should be placed after that Term. The notation is used to concentrate in one place all the Terms that are composed of, e.g., the word Attitude preceded by one of the words Social or Spatial.

Term	Definition
Acceleration	The 2nd order time derivative of a Position or Orientation.
- *Coordinate*	The acceleration measured in a coordinate system.
- *Proper*	The physical acceleration, i.e., measured by an accelerometer experienced by an object.
Accuracy	An estimate of how well the measurement of a physical entity corresponds to the actual value of that entity.
Alert	A Data Type representing environment-related elements that should be treated with priority by the Traffic Obstacle Avoidance AIM.
AMS-MAS Message	A Data Type representing the command issued by the Autonomous Motion Subsystem instructing the Motion Actuation Subsystem to change the Ego CAV's Spatial Attitude SA_A at time t_A to Spatial Attitude SA_B at time t_B and the MAS response the to the AMS-MAS Message.
AMS-HCI Message	A Data Type representing high-level instructions issued by HCI to AMS to request it to reach a destination and the AMS response.
Brake	A system activated by the Motion Actuation Subsystem having the function to decelerate a CAV.
- Command	A Data Type representing the command that the Motion Actuation Subsystem issues to the Brakes after interpreting an AMS-MAS Message.
- Response	A Data Type representing the Brakes' response to the AMS Command Interpreter in response to a Brake Command.
Connected Autonomous Vehicle	(CAV) The information technology-related components of a vehicle enabling it to autonomously reach a destination by: 1. Understanding human utterances in the Subsystem. 2. Planning a Route. 3. Sensing and building a series of Basic Environment Descriptors (BED).

	4. Exchanging refined BEDs (FED) with other CAVs and CAV-Aware entities. 5. Making decisions about how to execute the Route. 6. Acting on the Motion Actuation Subsystem.
- Aware	An attribute of equipment possessing some of the sensing and communication capabilities of a CAV without being a CAV, e.g., Roadside Units and Traffic Lights.
- Centre	The point in a CAV selected to have coordinates (0,0,0).
- *Ego*	The Object in the Environment that the CAV recognises as being itself.
- Environment	The Digital Representation of the portion of the external environment of current interest to a CAV.
- Identifier	A Data Type uniquely identifying a CAV and carrying information, such as: 1. Country code where the CAV has been registered. 2. Registration number in that country. 3. CAV manufacturer identifier. 4. CAV model identifier.
- State	A Description of the state of the CAV generated by the CAV's AMS using information available inside the CAV as assessed by the CAV and received from an external source, e.g., another CAV or Roadside Unit.
Data	Information in digital form
- *Accelerometer*	A Data Type representing the acceleration forces acting on a CAV.
- *Environment*	A Data Type representing the Environment such as produced by an Environment Sensing Technology or derived from the Basic or Full Environment Descriptors.
- *LiDAR*	A Data Type representing Data captured by a LiDAR sensor.
- *Odometer*	A Data Type representing the distance from an initial to the current Position measured by the number of wheel rotations times the tire circumference (π x tire diameter).
- *Offline Map*	A Data Type representing the type of Data provided in response to a given set of coordinate values.
- *RADAR*	A Data Type representing the Data captured by a RADAR sensor.
- *Spatial*	A Data Type containing Odometer, Speedometer, Accelerometer, and Inclinometer Data.
- *Speedometer*	A Data Type representing the speed of a CAV as measured by the sensor.
- *Ultrasound*	A Data Type representing the Data provided by an ultrasonic sensor.
- *Weather*	Weather Data is a set of data that includes Temperature, Humidity, Air Pressure, Ice conditions, Wind conditions and water in various states.
Decision Horizon	The time interval within which a decision is planned to be implemented.
Sensing Technology	(EST) One of the technologies used to sense the environment by the Environment Sensing Subsystem, e.g., Audio, LiDAR, RADAR, Ultrasound, Visual including the Offline Map.
Environment Scene Descriptors	A Data Type representing the combination of EST-specific Scene Descriptors (e.g., 2D, 2.5D, or 3D) used by an EST Scene Description in an EST-specific time window.

- *Basic*	(BED) A Data Type representing Environment using information provided by a variety of sensors and including the Scene Description produced by integrating the available Environment Sensing Technologies and Weather Data.
- *Full*	(FED) the Environment Descriptors that extend the Basic Environment Descriptors of the Ego CAV with elements provided by other CAVs in range and CAV-Aware entities, the CAV State and the Road State.
Global Navigation Satellite System (GNSS)	A Data Type provided by one of the global navigation systems such as GPS, Galileo, Glonass, BeiDou, Quasi Zenith Satellite System (QZSS) and Indian Regional Navigation Satellite System (IRNSS).
- Object	A Data Type composed of GNSS Data and GNSS Qualifier.
- Qualifier	A Data Type providing information of GNSS Data, such as Sub-Type, Format, and Attributes.
Goal	The Spatial Attitude planned to be reached at the end of a Decision Horizon.
Inertial Measurement Unit	An inertial positioning device, e.g., odometer, accelerometer, speedometer, inclinometer, etc.
Latitude	A Data Type representing the angular distance of a point on the surface of the Earth placed North or South of the equator measured in degrees.
LiDAR	A Data Type representing signals captured by an active time-of-flight sensor operating in the µm range – ultraviolet, visible, or near infrared light (900 to 1550 nm).
- Object	A Data Type composed of LiDAR Data and LiDAR Qualifier.
- Qualifier	A Data Type providing information of LiDAR Data, such as Sub-Type, Format, and Attributes.
Longitude	A Data Type representing the angular distance of a point west of the Greenwich meridian measured in degrees.
MAS Subsystem	The CAV Subsystem interpreting AMS-MAS Messages from the AMS; issuing commands and receiving responses from Brakes, Wheel, and Motors; and responding with AMS-MAS Messages to the AMS.
Motor	A system activated by the Motion Actuation Subsystem having the function to accelerate a CAV.
- Command	A Data Type representing the command issued to a Motor by the Motion Actuation Subsystem after interpreting an AMS-MAS Message.
- Response	A Data Type representing the Motor's response to AMS Command Interpreter in response to a Motor Command.
Offline Map	A previously created digital map of an Environment and associated metadata.
- Object	A Data Type composed of Offline Map Data and Offline Map Qualifier.
- Qualifier	A Data Type providing information of Offline Map Data, such as Sub-Type, Format, and Attributes.
Pose	A Data Type representing the Point of View of the CAV as obtained by processing the data from the CAV sensors.
RADAR	A Data Type representing signals captured by an active time-of-flight sensor operating in the 24-81 GHz range.

- Object	A Data Type composed of RADAR Data and RADAR Qualifier.
- Qualifier	A Data Type providing information of RADAR Data, such as Sub-Type, Format, and Attributes.
Remote	
- AMS	The Autonomous Motion Subsystem of a CAV or CAV-Aware entity in range.
- HCI	The Human-CAV Interaction Subsystem of a CAV or CAV-aware entity in range.
Road	A portion of the Environment typically used by CAVs for their movements.
- Geometry	A Data Type representing the positioning of the physical elements of the roadway, e.g., traffic poles, road signs, traffic lights, etc.
- State	A Data Type representing the state of the road the CAV is traversing such as weather, submersion, destruction, pothole and roadwork position, etc.
Roadside Unit	A wireless communicating device located on the roadside providing information to CAVs in range.
Route	A Data Type representing a sequence of Waypoints.
Scene Descriptors	
- *LiDAR*	A Data Type describing the LiDAR Data and produced by the LiDAR Scene Description AIM also using previous Basic Environment Representations.
- *Offline Map*	A Data Type including the objects of a Scene described by an Offline Map.
- *RADAR*	A Data Type representing the Visual Data captured by RADAR and produced by the RADAR Scene Description AIM also using previous Basic Environment Representations.
- *Ultrasound*	A Data Type representing the Visual Data captured by Ultrasound and produced by the Ultrasound Scene Description AIM also using previous Basic Environment Representations.
Shape	A Data Type representing the volume occupied by a CAV.
Subsystem	One of HCI, ESS, AMS, and MAS.
Traffic	
- Rules	The Digital Representation of the traffic rules applying to an Environment.
- Signals	A Data Type representing the traffic signals on a road and around it, their Spatial Attributes, and the semantics of the traffic signals.
Ultrasound	A Data Type representing signals captured by an ultrasonic sensor, an active time-of-flight sensor typically operating in the 40 kHz to 250 kHz range.
- Object	A Data Type composed of Ultrasound Data and Ultrasound Qualifier.
- Qualifier	A Data Type providing information of Ultrasound Data, such as Sub-Type, Format, and Attributes.
Waypoint	The coordinates of a point on an Offline Map.
Wheel	A system activated by the Motion Actuation Subsystem having the function to rotate a CAV.

- Command	A Data Type representing the command issued to the Steering Wheel by the Motion Actuation Subsystem after interpreting an AMS-MAS Message.
- Response	A Data Type representing the Wheel's response to the AMS Command Interpreter in response to a Direction Command.

5 References

5.1 Normative References

1. MPAI: Technical Specification: AI Framework (MPAI-AIF) V2.1,
2. MPAI; Technical Specification: Context-based Audio Enhancement (MPAI-CAE) – Use Cases (CAE-USC) V2.3.
3. MPAI; Technical Specification: Human and Machine Communication (MPAI-HMC) V2.0.
4. MPAI; Technical Specification: Technical Specification: Multimodal Conversation (MPAI-MMC) V2.3.
5. MPAI; Technical Specification: Technical Specification: MPAI Metaverse Model (MPAI-MMM) – Technologies (MPAI-MMM) V2,0
6. MPAI; Technical Specification: Technical Specification: Object and Scene Description (MPAI-OSD) V1.3.
7. MPAI; Technical Specification: Technical Specification: Portable Avatar Format (MPAI-PAF) V1.4.
8. MPAI; Technical Specification: Technical Specification: AI Module Profiles (MPAI-PRF) V1.0.

5.2 Informative References

9. MPAI Statutes
10. MPAI Patent Policy
11. MPAI Technical Specifications
12. MPAI; Technical Specification: Governance of the MPAI Ecosystem (MPAI-GME) V1.1.

6 Architecture and Operation

6.1 Introduction

The Connected Autonomous Vehicle (CAV) specified by CAV-TEC is a system able to instruct a vehicle with at least three wheels to reach a Destination from a current Pose at the request of a human or a process respecting the local traffic law, exploiting information that is captured and processed by the CAV and communicated by other CAVs. Figure 1 represents an example of the type of environment that a CAV is requested to traverse and Figure 2 depicts the four subsystems of which a CAV is composed, although this partitioning is not a functional requirement as components of a subsystem may be located in another subsystem, provided the interfaces specified by CAV-TEC are preserved.

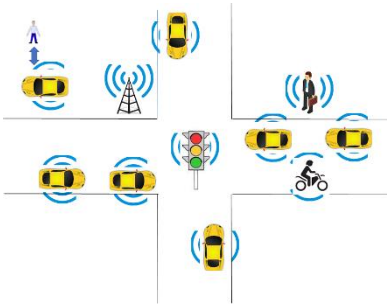

Figure 1 - An example of an environment traversed by a CAV

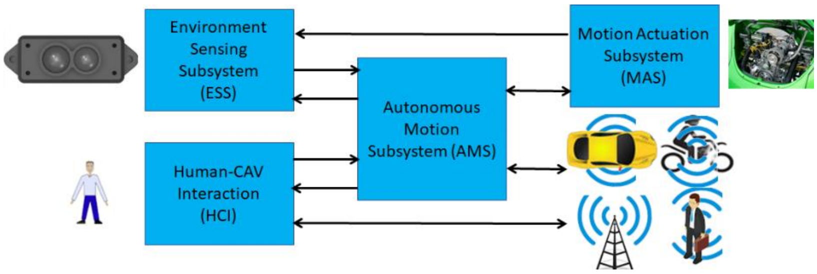

Figure 2 - The subsystems of a CAV

In Figure 2, a human approaches a CAV and requests the Human-CAV Interaction Subsystem (HCI) to be taken to a destination using a combination of four media – Text, Speech, Face, and Gesture. Alternatively, a remote process may make a similar request to the CAV.

Either request is passed to the Autonomous Motion Subsystem (AMS), which requests the Environment Sensing Subsystem (ESS) to provide the current CAV Pose. With this information from ESS (current Pose), the Destination, and access to Offline Maps, the AMS can propose one or more Routes, one of which the human or process can select.

With the human aboard, the AMS continues to receive environment information from the ESS - possibly complemented with information received from other CAVs in range - and instructs the Motion Actuation Subsystem to make appropriate motions.

6.2 Human-CAV Interaction

The operation of the HCI in its interaction with humans is best explained using the CAV-HCI Reference Model of Figure 3.

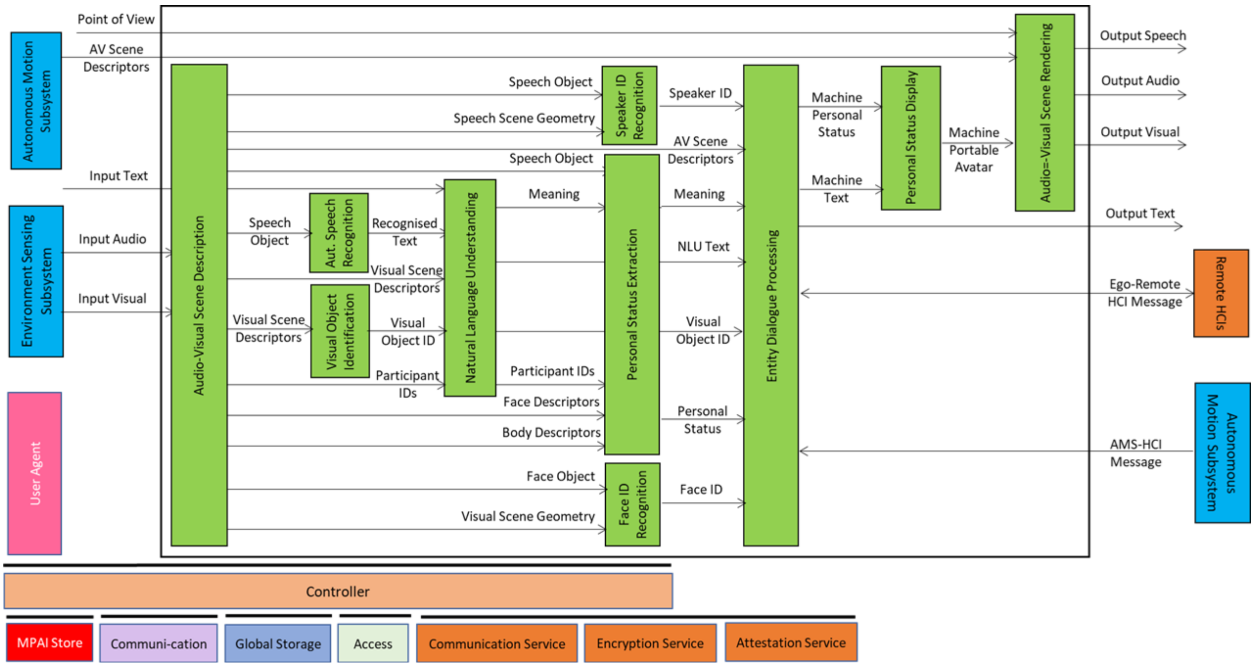

Figure 3 - Reference Model of CAV-HCI

The Audio-Visual Scene Description (AVS) monitors the environment and produces Audio-Visual Scene Descriptors from which it extracts Speech Scene Descriptors and from these, Speech Objects corresponding to any speaking humans in the environment surrounding the CAV. Visual

Scene Descriptors may also be extracted in the form of Face and Body Descriptors of all humans present.

The CAV activates Automatic Speech Recognition (ASR) to have the speech of each human recognised and converted into Recognised Text. Each Speech Object is identified according to their position in space. The CAV also activates the Visual Object Identification (VOI) that is able to produce the Instance IDs of Visual Objects as indicated by humans.
Natural Language Understanding (NLU) processes the Speech Objects, produces Refined Text, and extracts Meaning from the Text of each input Speech. This process is facilitated by the use of the IDs of the Visual Objects provided by VOI.

Speaker Identity Recognition (SIR) and Face Identity Recognition (FIR) help the CAV to reliably obtain the Identifiers of the the humans the HCI is interacting with. If the Face ID(s) provided by FIR correspond to the ID(s) provided by SIR, the CAV may proceed to attend to further requests. Especially with humans aboard, Personal Status Extraction (PSE) provides useful information regarding the humans' state of mind by extracting their Personal Status.
The CAV interacts with humans through Entity Dialogue Processing (EDP). When a human requests to be taken to a Destination, the EDP interprets and communicates the request to the Autonomous Motion Subsystem (AMS). A dialogue may then ensue where the AMS may offer different choices to satisfy potentially different human needs (e.g., a long but comfortable Route or short but less predictable).

Then, while the CAV moves to the Destination, the HCI may have a conversation with the humans, show the Full Environment Descriptors developed by the AMS to the passengers, and may communicate information about the CAV from the Ego AMS or more generally from the HCIs of remote CAVs.

The HCI responds using the two main outputs of the EDP: Text and Personal Status. These are used by the Personal Status Display (PSD) to produce the Portable Avatar of the HCI conveying Speech, Face, and Gesture synthesised to render the HCI Text and Personal Status. Audio-Visual Scene Rendering (AVR) renders Audio, Speech, and Visual information using the HCI Portable Avatar. Alternatively, it can display the AMS's Full Environment Descriptors from the Point of View selected by the human.

The HCI interacts with passengers in several ways:

1. By responding to commands/queries from one or more humans at the same time, e.g.:
    1. Commands to go to a waypoint, park at a place, etc.
    2. Commands with an effect in the cabin, e.g., turn off air conditioning, turn on the radio, call a person, open a window or door, search for information, etc.
2. By conversing with and responding to questions from one or more humans at the same time about travel-related issues, e.g.:
    1. Humans request information, e.g., time to destination, route conditions, weather at destination, etc.
    2. Humans ask questions about objects in the cabin.
3. By following the conversation on travel matters held by humans in the cabin if
    1. The passengers allow the HCI to do so, and
    2. The processing is carried out privately inside the CAV.

6.3 Environment Sensing Subsystem

The operation of the Environment Sensing Subsystem (ESS) is best explained using the Reference Model of the CAV-ESS subsystem depicted in Figure 4.

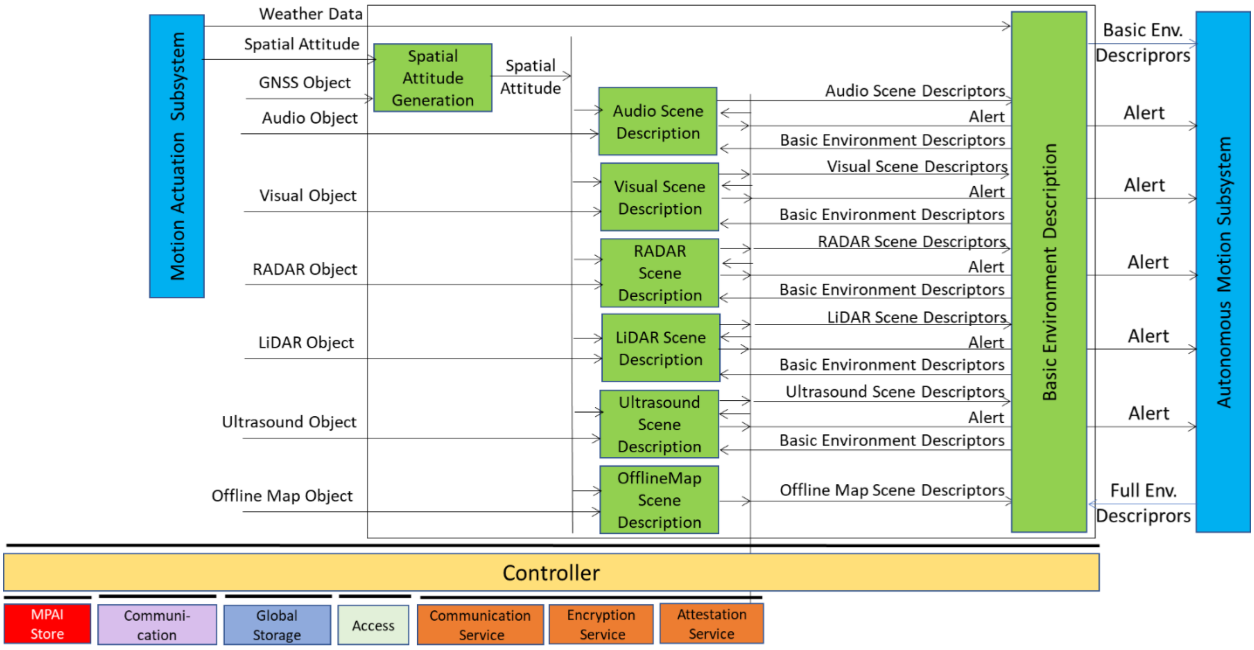

Figure 4 - Reference Model of CAV-ESS

When the CAV is activated in response to a request by a human owner or renter or by a process, Spatial Attitude Generation continuously computes the CAV's Spatial Attitude relying on the initial Motion Actuation Subsystem's Spatial Attitude, and information from the Global Navigation Satellite Systems (GNSS), if available.

An ESS may be equipped with a variety of Environment Sensing Technologies (EST). CAV-TEC assumes they are (but not required to all be supported by an ESS implementation) Audio, LiDAR, RADAR, Ultrasound, and Visual. Offline Map is considered as an EST.

An EST-specific Scene Description receives EST-specific Data Objects, produces EST specific Scene Descriptors which are integrated into the Basic Environment Descriptors (BED) by the Basic Environment Description using all available sensing technologies, Weather Data, Road State, and possibly the Full Environment Descriptors of previous instants provided by the AMS. Note that, although in Figure 4 each sensing technology is processed by an individual EST, an implementation may combine two or more Scene Description AIMs to handle two or more ESTs, provided the relevant interfaces are preserved. An EST-specific Scene Description may need to access the BED of previous instants and may produce Alerts that are immediately communicated to AMS.

The Objects in the BEDs may carry Annotations specifically related to traffic signalling, e.g.: Position and Orientation of traffic signals in the environment, Traffic Policemen, Road signs (lanes, turn right/left on the road, one way, stop signs, words painted on the road), Traffic signs – vertical signalisation (signs above the road, signs on objects, poles with signs), Traffic lights, Walkways, and Traffic sounds (siren, whistle, horn).

6.4 Autonomous Motion Subsystem

The operation of the Autonomous Motion Subsystem (AMS) is best explained using the Reference Model of the CAV-AMS subsystem depicted in Figure 5.

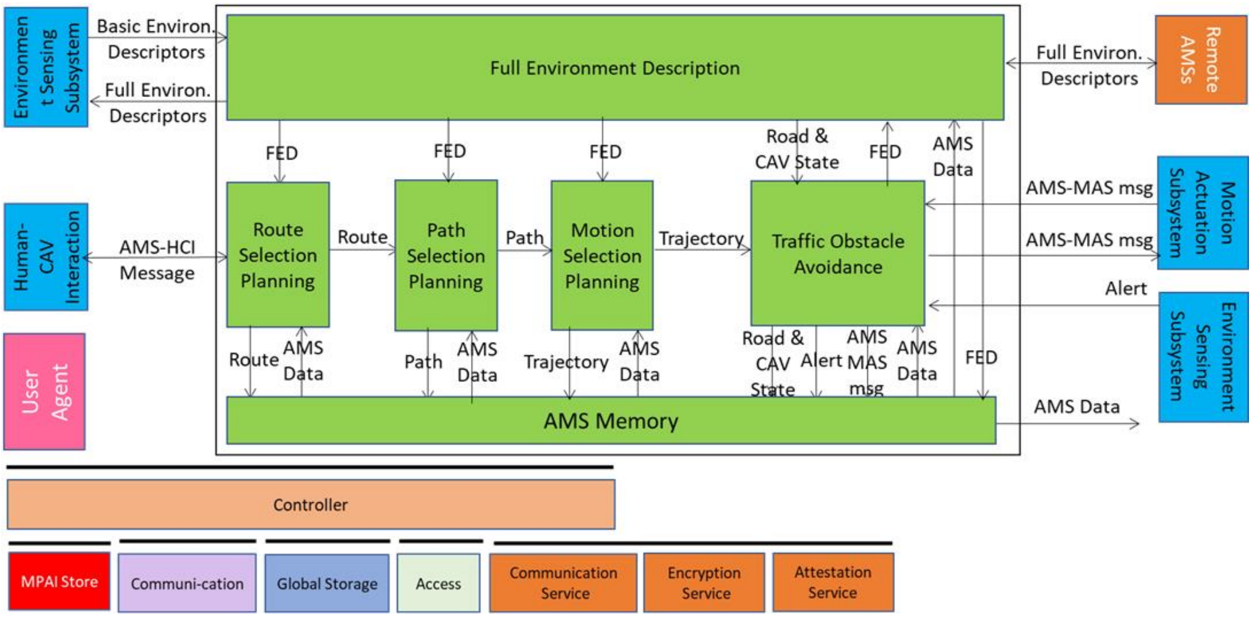

Figure 5 - Reference Model of CAV-AMS

When the HCI sends the AMS a request of a human or a process to move the CAV to a Destination, Route Planning uses the Basic Scene Descriptors from the ESS and produces a set of Waypoints starting from the current Pose up to the Destination.

When the CAV is in motion, Route Planning causes Path Selection Planning to generate a set of Poses to reach the next Waypoint. Full Environment Description may request the AMSs of Remote CAVs to send (subsets of) their Scene Descriptors and integrates all sources of Environment Descriptors into its Full Environment Descriptors (FED) and may also respond to similar requests from Remote CAVs.

Motion Selection Planning generates a Trajectory to reach the next Pose in each Path. Traffic Obstacle Avoidance receives the Trajectory and checks if any Alert was received that would cause a collision with the current Trajectory. If a potential collision is detected, Traffic Obstacle Avoidance requests a new Trajectory from Motion Planner, otherwise Traffic Obstacle Avoidance issues an AMS-MAS Message to Motion Actuation Subsystem (MAS).

The MAS sends an AMS-MAS Message to AMS informing it about the execution of the AMS-MAS Message received. The AMS, based on the received AMS-MAS Messages, may discontinue the execution of the earlier AMS-MAS Message, issue a new AMS-MAS Message, and inform Traffic Obstacle Avoidance. The decision of each element of the chain may be recorded in the AMS Memory (“black box”).

6.5 Motion Actuation Subsystem

The operation of the Motion Actuation Subsystem (MAS) is best explained using the Reference Model of the CAV-MAS subsystem depicted in Figure 6.

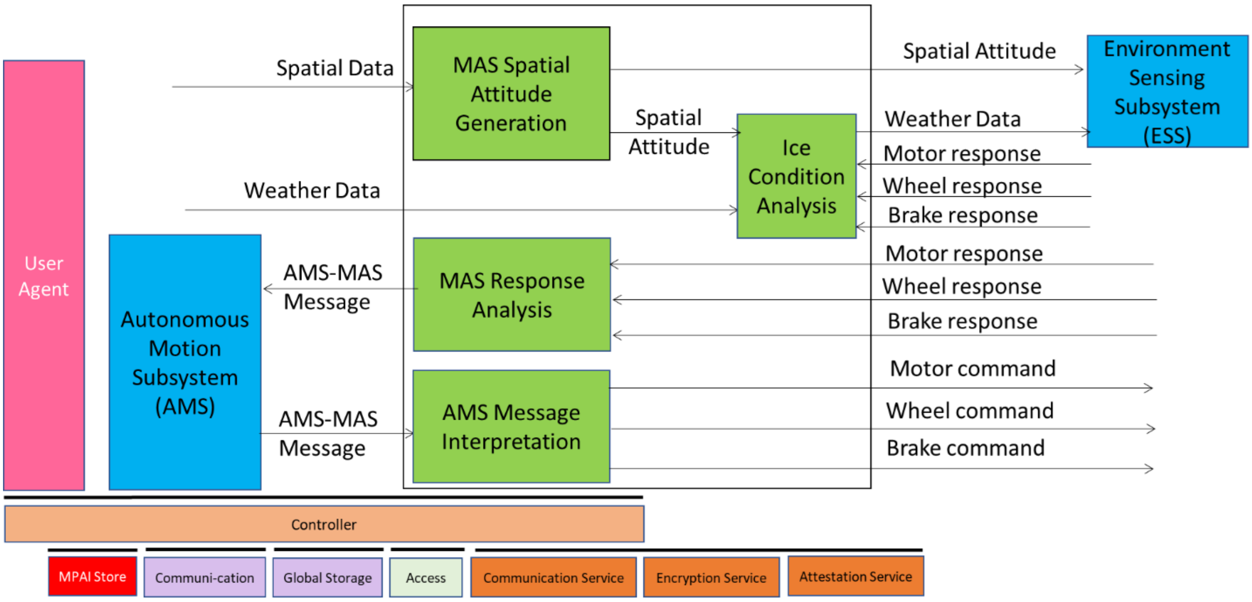

Figure 6 - Reference Model of CAV-AMS

When the AMS Message Interpretation receives the AMS-MAS Message from the AMS, it interprets the Messages, partitions it into commands, and sends them to the Brake, Motor, and Wheel mechanical subsystems. CAV-TEC is silent on how the three mechanical subsystems process the commands but specifies the format of the commands issued to and received by AMS Message Interpretation. The result of the interpretation is sent as an AMS-MAS Message to AMS.

MAS includes two more AIMs. Spatial Attitude Generation computes the initial Ego CAV's Spatial Attitude using the Spatial Data provided by Odometer, Speedometer, Accelerometer, and Inclinometer. This initial Spatial Attitude is sent to the ESS. Ice Condition Analysis augments the Weather Data by analysing the Brake, Motor, and Wheel mechanical subsystems' responses and sends the augmented Weather Data to the E

7 Reference Model

7.1 Functions

A Connected Autonomous Vehicle is a physical system that:

1. Converses with humans by understanding their utterances, e.g., "take me home" or "show me the environment you see".
2. Senses the environment where it is located or traverses. *Figure 1* is an *example* of the environments targeted CAV-TEC.
3. Plans a Route enabling the CAV to reach the requested destination.
4. Autonomously reaches the destination by:
    1. Building digital representations of the environment.
    2. Moving in the physical environment.
    3. Exchanging elements of such Environment Representations with other CAVs and CAV-aware entities.
    4. Making decisions about how to execute the Route.
    5. Actuating the CAV motion to implement the decisions.

7.2 Reference Architecture

The MPAI-CAV Reference Model is composed of four Subsystems depicted in Figure 1 and implemented as AI Workflows:

1. Human-CAV Interaction (HCI)
2. Environment Sensing Subsystem (ESS)
3. Autonomous Motion Subsystem (AMS)
4. Motion Actuation Subsystem (MAS)

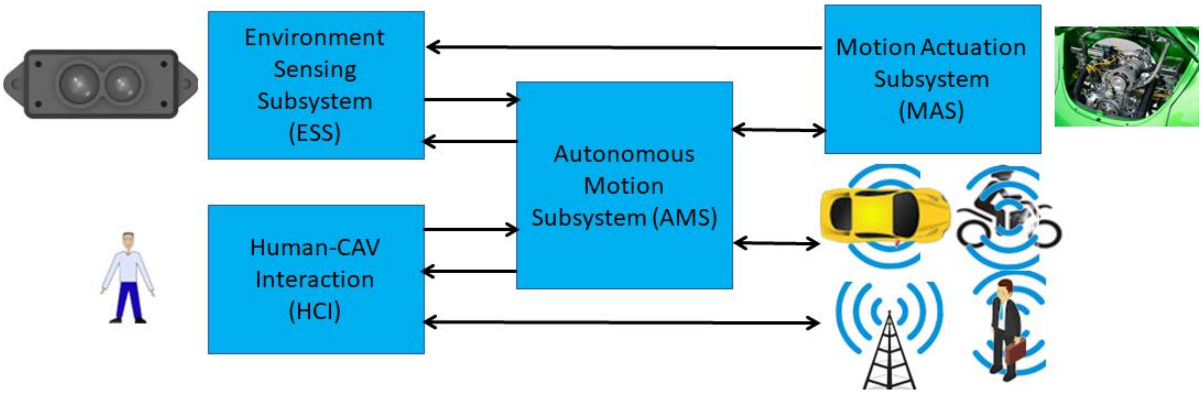

Figure 1 – The MPAI-CAV subsystems

The operation of a CAV unfolds according to the following workflow, which is not an exhaustive description of all the functions performed:

Table 1 - High-level CAV operation

Entity	Action
Human	Requests the CAV, via HCI, to take the human to a destination.
HCI	1. Authenticates humans. 2. Interprets the request of humans. 3. Issues commands to the AMS.
AMS	1. Requests ESS to provide the current Pose.
ESS	1. Computes and sends the Basic Environment Descriptors (BED) to AMS.
AMS	1. Computes and sends Route(s) to HCI.
HCI	1. Sends travel options to Human.
Human	1. May integrate/correct their instructions. 2. Issues commands to HCI.
HCI	1. Communicates Route selection to AMS.
AMS	1. Sends the BED to the *AMS*s of other CAVs. 2. Computes the Full Environment Descriptors (FED). 3. Decides best motion to reach the destination. 4. Issues appropriate commands to *MAS*.
MAS	1. Executes the Command. 2. Sends response to *AMS*.
Human	1. Interacts and holds conversation with other humans on board and the *HCI*. 2. Issues commands to *HCI*. 3. Requests *HCI* to render the FED.

4. Navigates the FED.
5. Interacts with humans in other CAVs.

HCI Communicates with *HCI*s of Remote CAVs on matters related to human passengers.

7.3 I/O Data

Table 2 gives the input/output data of the Connected Autonomous Vehicle.

Table 2 - I/O data of Connected Autonomous Vehicle

Input data	From	Description
Audio Object	Environment	Environment Data captured by Microphones with Qualifier.
Brake Response	Brakes	Acts on brakes, gives feedback.
Ego-Remote AMS Message	Ego AMS	Message to Remote AMS.
Ego-Remote HCI Message	Ego HCI	Message to Remote HCI.
GNSS Object	~1 & 1.5 GHz Radio	Data from various Global Navigation Satellite System (GNSS) sources with Qualifier.
LiDAR Object	Environment	Environment Data captured by LiDAR with Qualifier.
Motor Response	Wheel Motor	Forces wheels rotation, gives feedback.
RADAR Object	Environment	Environment Data captured by RADAR with Qualifier.
Text Object	Cabin Passengers	Text complementing/replacing User input.
Ultrasound Object	Environment	Environment Data captured by Ultrasound with Qualifier.
Visual Object	Environment	Environment Data captured by cameras with Qualifier.
Weather Data	Environment	Temperature, Air pressure, Humidity, etc.
Wheel Response	Steering Wheel	Moves wheels by an angle, gives feedback.
Output data	**To**	**Description**
AMS Data	Outside device	AMS Data stored in AMS Memory provided for analysis.
Audio Object	Cabin Passengers	HCI Response, Rendered Full Environment Descriptors.
Brake Command	Brakes	Acts on Brakes.
Ego-Remote AMS Message	Remote AMS	Message from Ego AMS to Remote AMS.
Ego-Remote HCI Message	Remote HCI	Message from Ego HCI to Remote HCI.
Motor Command	Wheel Motors	Activates/suspends/reverses wheel rotation.

Text Object	Cabin Passengers	Text from HCI.
Visual Object	Cabin Passengers	Environment as seen by CAV and/or HCI rendering.
Wheel Command	Wheel	Moves wheel by an angle.

7.4 Functions of AI Workflows

Table 3 describes the high-level functions of all CAV AI Workflows.

Table 3 – Functions of CAV AI Workflows

AIW	Function
Human-CAV Interaction	Recognises human owner/renter, responds to humans' commands and queries, converses with humans, manifests itself as a perceptible entity, exchanges information with the Autonomous Motion Subsystem in response to humans' requests, and communicates with other CAVs or CAV-Aware entities.
Environment Sensing Subsystem	Senses the environment's Electromagnetic and Acoustic information, receives Ego CAV's Spatial Attitude and Weather Data from own ESS, requests location-specific Data from Offline Map(s), produces the best estimate of the Ego CAV Spatial Attitude, sensor-specific Scene Descriptors and Alerts to AMS, Basic Environment Descriptors (BED), passes the BEDs to HCI and AMS), and requests/receives elements of the Full Environment Descriptors (FED) to/from Remote AMSs.
Autonomous Motion Subsystem	Converses with HCI (and HCI with humans) to provide a Route, requests and provides FED subsets to selected Remote CAVs, produces FED, generates Paths, Trajectory, checks Trajectory implementation considering Alerts from ESS's technology-specific Scene Descriptions, issues commands to and processes responses from MAS, stores Data received/produced in AMS Memory.
Motion Actuation Subsystem	Transmits Weather Data and Spatial Data-based Spatial Attitude of the CAV to ESS, receives AMS-MAS Messages from AMS, translates AMS-MAS Message into Brake, Motor, and Wheel Commands, packages and sends Brake, Motor, and Wheel Responses from its Brakes, Motors, and Wheel to AMS.

7.5 I/O Data of AI Workflows

Table 4 gives the AI Modules of the Human-CAV Interaction depicted in Figure 1.

Table 4 – AI Workflows of Connected Autonomous Vehicle

AIW	Input	Output
Human-CAV Interaction	Point of View AMS-HCI Message Audio-Visual Scene Descriptors Ego-Remote HCI Message Text Object	AMS-HCI Message Ego-Remote HCI Message Text Object Speech Object Audio Object Visual Object

	Audio Object Visual Object	
Environment Sensing Subsystem	Audio Object GNSS Object LiDAR Object Offline Map Object RADAR Object Ultrasound Object Visual Object Weather Data Spatial Attitude Full Environment Descriptors	Alert Basic Environment Descriptors
Autonomous Motion Subsystem	Alert AMS-HCI Message AMS-MAS Message Basic Environment Descriptors Full Environment Descriptors Ego-Remote AMS Message	AMS-HCI Message AMS-MAS Message Full Environment Descriptors Ego-Remote AMS Message AMS Data
Motion Actuation Subsystem	AMS-MAS Message Spatial Data Brake Response Motor Response Wheel Response Weather Data	AMS-MAS Message Brake Command Motor Command Wheel Command Weather Data Spatial Attitude

7.6 AIWs and JSON Metadata

Table 5 provides the links to the AIW specifications and to the JSON Metadata.

Table 5 – AIWs and JSON Metadata

AIW	Name	JSON
MMC-HCI	Human-CAV Interaction	X
CAV-ESS	Environment Sensing Subsystem	X
CAV-AMS	Autonomous Motion Subsystem	X
CAS-MAS	Motion Actuation Subsystem	X

8 AI Workflows

8.1 Human-CAV Interaction

8.1.1 Functions

The Human-CAV interaction (HCI) Subsystem has the function to recognise the human owner or renter, respond to humans' commands and queries, converse with humans, manifests itself as a perceptible entity, exchange information with the Autonomous Motion Subsystem in response to humans' requests, and communicate with HCIs on board other CAVs.

8.1.2 Reference Model

Figure 1 represents the Human-CAV Interaction (HCI) Reference Model.

It is assumed that Natural Language Understanding produces a Refined Text that is either the refined Recognised Text or the direct Input Text, depending on which one is being used. Meaning is always computed based on the available Text - Refined or Input.

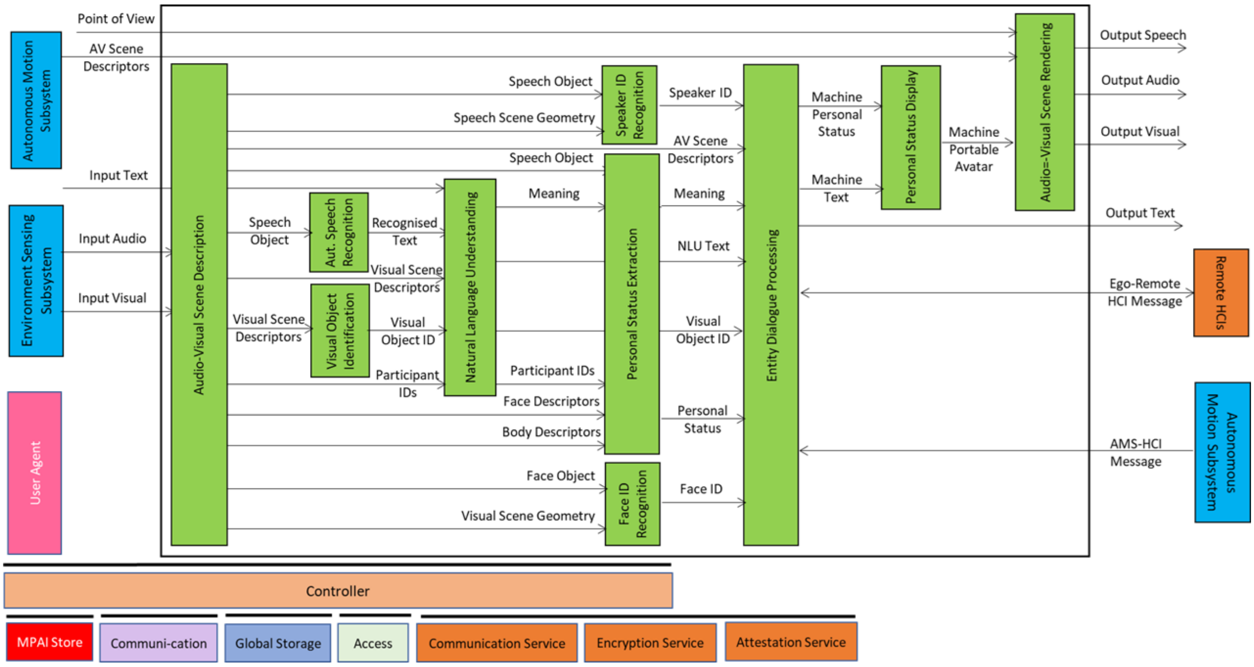

Figure 1 – Human-CAV Interaction Reference Model

The operation of the HCI subsystem is described by the following scenario where a group of humans approaches the CAV outside the CAV or is sitting inside the CAV:

1. Audio-Visual Scene Description (AVS) produces:
   1. Speech Scene Descriptors in the form of Speech Objects corresponding to each speaking human in the Environment (outside or inside the CAV)..
   2. Visual Scene Descriptors in the form of Descriptors of Faces and Bodies.
   3. All non-Speech Objects are removed from or signalled in the Audio Scene.
2. Automatic Speech Recognition (ASR) recognises the speech of each human and produces Recognised Text supporting multiple Speech Objects as input properly identified by the Spatial Attitude.
3. Visual Object Identification (VOI) produces Instance IDs of Visual Objects indicated by humans.
4. Natural Language Understanding (NLU)produces Refined Text and extracts Meaning from the Recognised Text of each Input Speech using the spatial information of Visual Object Identifiers.
5. Speaker Identity Recognition (SIR) and Face Identity Recognition (FIR) identifies the humans the HCI is interacting with. If FIR provides Face IDs corresponding to the Speaker IDs, Entity Dialogue Processing AIM can correctly associate the Speaker IDs (and the corresponding Text) with the Face IDs.
6. Personal Status Extraction (PSE) extracts the Personal Status of the humans.
7. Entity Dialogue Processing (EDP)
   1. Communicates with the Autonomous Motion Subsystem of the Ego CAV to request to:
      1. Move the CAV to a destination.

    2. Views the Full Environment Descriptors for the passengers' benefit.
    3. Be informed about CAV's situation.
    4. Receive relevant information for passengers.
  2. Communicates with the Autonomous Motion Subsystems of Remote CAVs.
  3. Produces the Machine Text and Machine Personal Status.
8. Personal Status Display (PSD) produces the Machine Portable Avatar conveying Machine Speech, Machine Personal Status, and any other information that may be relevant for the the Audio-Visual Rendering AIM .
9. Audio-Visual Scene Rendering (AVR) renders Audio, and Visual information using Machine Portable Avatar or the Autonomous Motion Subsystem's Full Environment Descriptors based on the Point of View provided by the human.
10. Entity Dialogue Processing (EDP)
    1. Requests the AMS subsystem to provide candidate Routes in response to a human requesting to be taken to a destination.
    2. Responses from AMS are processed by EDP and converted to multimodal messages understandable by the human.
    3. Eventually, the human accepts the Route or further elaborates on the EDP response.
    4. May receive messages from Ego AMS or Remote HCI that are processed and converted to multimodal messages understandable by the human.

The HCI interacts with the humans in the cabin in several ways:

1. By responding to commands/queries from one or more humans at the same time, e.g.:
    1. Commands to go to a waypoint, park at a place, etc.
    2. Commands with an effect in the cabin, e.g., turn off air conditioning, turn on the radio, call a person, open window or door, search for information etc.
2. By conversing with and responding to questions from one or more humans at the same time about travel-related issues (in-depth domain-specific conversation), e.g.:
    1. Humans request information, e.g., time to destination, route conditions, weather at destination, etc.
    2. CAV offers alternatives to humans, e.g., long but safe way, short but likely to have interruptions.
    3. Humans ask questions about objects in the cabin.
3. By following the conversation on travel matters held by humans in the cabin if
    1. The passengers allow the HCI to do so, and
    2. The processing is carried out inside the CAV.

8.1.3 I/O Data

Table 1 gives the input/output data of Human-CAV Interaction. I/O Data to/from Remote HCI and Ego AMS are not part of this Technical Specification.

Table 1 - I/O data of Human-CAV Interaction

Input data	From	Comment
Point of View	Passenger	Passenger's Point of View looking at environment.
Audio-Visual Scene Descriptors	AMS Subsystem	Audio-Visual representation of the environment.
Input Audio	Environment, Passenger Cabin	User authentication, command/interaction with HCI, etc. and environment Audio.
Input Text	User	Text complementing/replacing User input

Input Visual	Environment, Passenger Cabin	Environment perception, User authentication, command/interaction with HCI, etc. and environment Visual.
AMS-HCI Message	AMS Subsystem	AMS response to HCI request.
Ego-Remote HCI Message	Remote HCI	Remote HCI to Ego HCI.
Output data	**To**	**Comment**
Output Text	Cabin Passengers	HCI's avatar Text.
Output Speech	Cabin Passengers	HCT's avatar Speech.
Output Audio	Cabin Passengers	HCI's avatar or FED Audio.
Output Visual	Cabin Passengers	HCI's avatar or FED Visual.
AMS-HCI Message	AMS Subsystem	HCI request to AMS, e.g., Route or Point of View.
Ego-Remote HCI Message	Remote HCI	Ego HCI to Remote HCI.

8.1.4 Functions of AI Modules

Table 2 gives the functions of all Human-CAV Interaction AIMs.

Table 2 – Functions of Human-CAV Interaction's AI Modules

AIM	Function
Audio-Visual Scene Description	1. Receives Audio and Visual Objects from the appropriate Devices. 2. Produces Audio-Visual Scene Descriptors.
Automatic Speech Recognition	1. Receives Speech Objects. 2. Produces Recognised Text.
Visual Object Identification	1. Receives Visual Scenes Descriptors. 2. Provides Instance ID of indicated Visual Object.
Natural Language Understanding	1. Receives Recognised Text. 2. Uses context information (e.g., Instance ID of object). 3. Produces Natural Language Understanding Text (using Refined or Input) and Meaning.
Speaker Identity Recognition	1. Receives Speech Object of a human and Speech Scene Geometry. 2. Produces Speaker ID.
Personal Status Extraction	1. Receives Speech Object, Meaning, Face Descriptors and Body Descriptors of a human with a Participant ID. 2. Produces the human's Personal Status.
Face Identity Recognition	1. Receives Face Object of a human and Visual Scene Geometry. 2. Produces Face ID.
Entity Dialogue Processing	1. Receives Speaker ID, Face ID, AV Scene Descriptors, Meaning, Natural Language Understanding Text , Visual Object ID, and Personal Status. Moreover it receives AMS-

	HCI Messages and Ego-Remote HCI Messages. 2. Produces Machine (HCI) Text Object and Personal Status. Moreover it produces AMS-HCI Messages and Ego-Remote HCI Messages.
Personal Status Display	1. Receives Machine Text Object and Machine Personal Status. 2. Produces Machine's Portable Avatar.
Audio-Visual Scene Rendering	1. Receives AV Scene Descriptors, Portable Avatar, and Point of View. 2. Produces Output Speech, Output Audio, and Output Visual.

8.1.5 I/O Data of AI Modules

Table 3 gives the AI Modules of the Human-CAV Interaction depicted in Figure 3.

Table 3 – AI Modules of Human-CAV Interaction AIW

AIM	Input	Output
Audio-Visual Scene Description	- Input Audio - Input Visual	- AV Scene Descriptors
Automatic Speech Recognition	- Speech Object	- Recognised Text
Visual Object Identification	- AV Scene Descriptors - Visual Objects	- Visual Object Instance ID
Natural Language Understanding	- Recognised Text - AV Scene Descriptors - Visual Object Instance ID - Input Text	- Natural Language Understanding Text - Meaning
Speaker Identity Recognition	- Speech Object - Speech Scene Geometry	- Speaker ID
Personal Status Extraction	- Meaning - Input Speech - Face Descriptors - Body Descriptors	- Personal Status
Face Identity Recognition	- Face Object - Visual Scene Geometry	- Face ID

Entity Dialogue Processing	- Ego-Remote HCI Message - AMS-HCI Message - Speaker ID - Meaning - Natural Language Understanding Text - Visual Object Instance ID - Personal Status - Face ID	- Ego-Remote HCI Message - AMS-HCI Message - Machine Text - Machine Personal Status
Personal Status Display	- Machine Personal Status - Machine Text	- Machine Portable Avatar
Audio-Visual Scene Rendering	- AV Scene Descriptors - Machine Portable Avatar - Point of View	- Output Text - Output Speech - Output Audio - Output Visual

8.1.6 AIW, AIMs and JSON Metadata

Table 4 provides the links to the AIW and AIM specifications and to the JSON syntaxes. AIMs/1 indicates that the column contains Composite AIMs and AIMs/2 indicates that the column contains Basic and Composite AIMs. AIMs/3 indicates the the column only contains Basic AIMs.

Table 4 – AIMs and JSON Metadata

AIW	AIMs/1	AIMs/2	AIMs/3	Name	JSON
MMC-HCI				Human-CAV Interaction	X
	OSD-AVS			Audio-Visual Scene Description	X
		CAE-ASD		Audio Scene Description	X
			CAE-AAT	Audio Analysis Transform	X
			CAE-ASL	Audio Source Localisation	X
			CAE-ASE	Audio Separation and Enhancement	X
			CAE-AST	Audio Synthesis Transform	X
			CAE-ADM	Audio Descriptors Multiplexing	X
		OSD-VSD		Visual Scene Description	X
	MMC-ASR			Automatic Speech Recognition	X
	OSD-AVA			Audio-Visual Alignment	X
	OSD-VOI			Visual Object Identification	X
		OSD-VDI		Visual Direction Identification	X
		OSD-VOE		Visual Object Extraction	X

	OSD-VII	Visual Instance Identification	X
MMC-NLU		Natural Language Understanding	X
MMC-SIR		Speaker Identity Recognition	X
MMC-PSE		Personal Status Extraction	**X**
	MMC-ETD	Entity Text Description	**X**
	MMC-ESD	Entity Speech Description	**X**
	PAF-EFD	Entity Face Description	**X**
	PAF-EBD	Entity Body Description	**X**
	MMC-PTI	PS-Text Interpretation	**X**
	MMC-PSI	PS-Speech Interpretation	**X**
	PAF-PFI	PS-Face Interpretation	**X**
	PAF-PGI	PS-Gesture Interpretation	**X**
	MMC-PMX	Personal Status Multiplexing	**X**
MMC-EDP		Entity Dialogue Processing	X
PAF-FIR		Face Identity Recognition	X
PAF-PSD		Personal Status Display	X
	MMC-TTS	Text-to-Speech	X
	PAF-IFD	Entity Face Description	X
	PAF-IBD	Entity Body Description	X
	PAF-PMX	Portable Avatar Multiplexing	X
PAF-AVR		Audio-Visual Scene Rendering	X

8.1.7 Conformance Testing

Table 5 provides the Conformance Testing Method for MMC-HCI AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 5 – Conformance Testing Method for MMC-HCI AIM

Receives	Input Audio	Shall validate against Audio Object Schema. Audio Data shall conform with Audio Qualifier.
	Input Text	Shall validate against Text Object Schema. Speech Data shall conform with Text Qualifier.
	Input Visual	Shall validate against Visual Object Schema. Speech Data shall conform with Visual Qualifier.
	AMS-HCI Message	Shall validate against AMS-HCI Message Schema.
	Ego-Remote HCI Message	Shall validate against Ego-Remote HCI Message Schema.

Produces Output Text	Shall validate against Text Object Schema. Text Data shall conform with Text Qualifier.
Output Speech	Shall validate against Speech Object Schema. Speech Data shall conform with Speech Qualifier.
Output Audio	Shall validate against Audio Object Schema. Audio Data shall conform with Audio Qualifier.
Output Visual	Shall validate against Visual Object Schema. Visual Data shall conform with Visual Qualifier.
AMS-HCI Message	Shall validate against AMS-HCI Message Schema.
Ego-Remote HCI Message	Shall validate against Ego-Remote HCI Message Schema.

8.2 Environment Sensing Subsystem

8.2.1 Functions

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

1. Senses the environment's
    1. Electromagnetic information from GNSS, LiDAR, RADAR, Visual sources.
    2. Acoustic information from Audio (16-20,000 Hz) and Ultrasound sources.
2. Receives, based on Data available at the Motion Actuation Subsystem,
    1. An estimate of the Ego CAV's Spatial Attitude.
    2. Weather information (e.g., temperature, pressure, humidity, etc.).
3. Requests location-specific Data from Offline Map(s).
4. Produces the best estimate of the Ego CAV Spatial Attitude by improving the location information received from MAS with GNSS information.
5. Produces EST-specific Scene Descriptors using Data stream from specific Environment Sensing Technologies (EST) on board the CAV (Audio, Visual, LiDAR, RADAR, Ultrasound, and Offline Map Data).
6. Produces a sequence of Basic Environment Descriptors. i.e., Scene Descriptors enhanced by additional information (BED) at a CAV-specific frequency by integrating the different EST-specific Scene Descriptors, Full Environment Descriptors at a previous time, and Weather Data.
7. Passes the BEDs to the Human-CAV Interaction (HCI) and Autonomous Motion (AMS) Subsystems.
8. Requests elements of the Full Environment Representations (FER) produced by AMS.

8.2.2 Reference Model

Figure 1 gives the Reference Model of the Environment Sensing Subsystem.

The sequence of operations of the Environment Sensing Subsystem unfolds as follows:

1. The Spatial Attitude Generation AIM computes the CAV's Spatial Attitude using the initial Motion Actuation Subsystem's Spatial Attitude and GNSS Object.
2. All EST-specific Scene Description AIMs available onboard:
    1. Receive EST-specific Data Objects, e.g., the RADAR Scene Descriptions AIM receives a RADAR Object provided by the RADAR EST (not shown in Figure 1). The Online Map is considered as an EST.
    2. Produce and send Alerts, if necessary, to the Autonomous Motion Subsystem.
    3. Accesses Basic Environment Descriptors of previous times, if needed.
    4. Produce EST-specific Scene Descriptors, e.g., the RADAR Scene Descriptors.
3. The Basic Environment Description AIM integrate the different EST-specific Scene Descriptors, Weather Data, and Road State into the Basic Environment Descriptors.

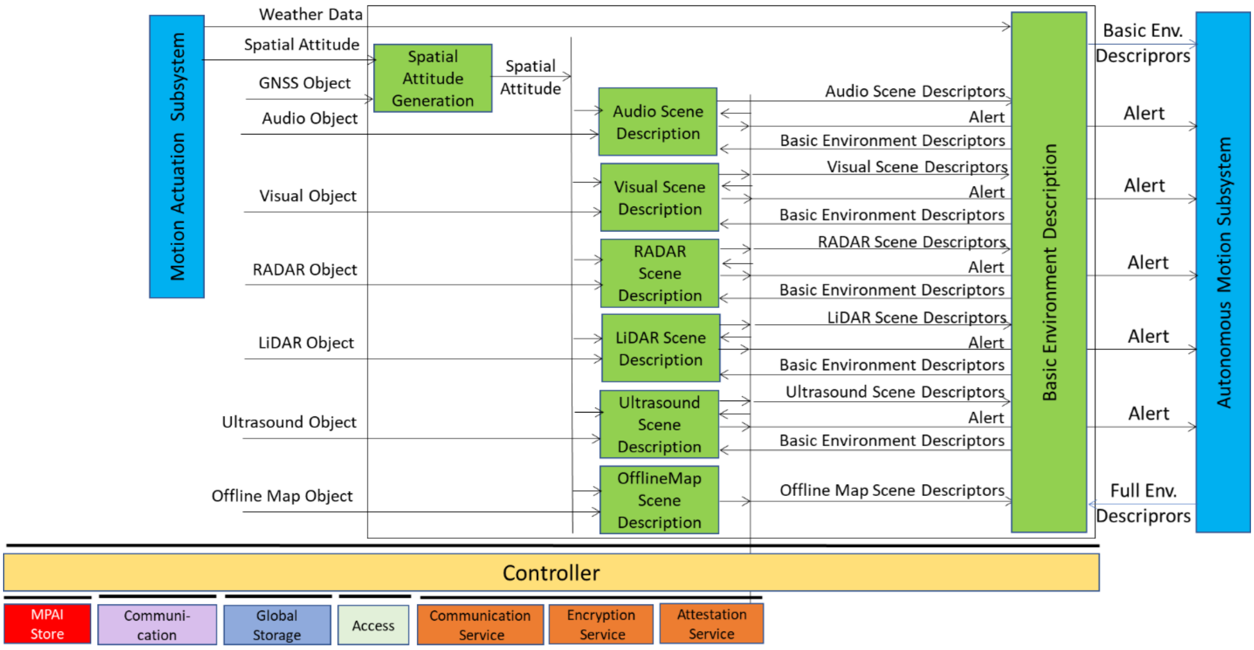

Figure 1 – Environment Sensing Subsystem Reference Model

Note 1: Although *Figure 1* shows individually processed ESTs, an implementation may combine two or more Scene Description AIMs to handle two or more ESTs, provided the relevant interfaces are preserved.

Note 2: The Objects in the BEDs may carry Annotations specifically related to traffic signalling, e.g.:

1. Position and Orientation of traffic signals in the environment:
2. Traffic Policemen
3. Road signs (lanes, turn right/left on the road, one way, stop signs, words on the road).
4. Traffic signs – vertical signalisation (signs above the road, signs on objects, poles with signs).
5. Traffic lights
6. Walkways
7. Traffic sound (siren, whistle, horn).

8.2.3 I/O Data

Table 1 gives the input/output data of the Environment Sensing Subsystem.

Table 1 - I/O data of Environment Sensing Subsystem

Input data	From	Comment
RADAR Object	~25 & 75 GHz Radio	Environment Capture with Radar
LiDAR Object	~200 THz infrared	Environment Capture with Lidar
Visual Object	Video (400-800 THz)	Environment Capture with visual cameras
Ultrasound Object	Audio (>20 kHz)	Environment Capture with Ultrasound
Offline Map Object	Local storage or online	cm-level data at time of capture
Audio Object	Audio (16 Hz-20 kHz)	Environment or cabin Capture with Microphone Array
GNSS Object	~1 & 1.5 GHz Radio	Get Pose from GNSS

Spatial Attitude	Motion Actuation Subsystem	To be fused with Pose from GNSS Data
Weather Data	Motion Actuation Subsystem	Temperature, Humidity, etc.
Full Environment Descriptors	Autonomous Motion Subsystem	FED refers to a previous time.
Output data	**To**	**Comment**
Alert	Autonomous Motion Subsystem	Critical information from an EST.
Basic Environment Descriptors	Autonomous Motion Subsystem	ESS-derived Environment Descriptors

8.2.4 Functions of AI Modules

Table 2 gives the functions of all AIMs of the Environment Sensing Subsystem.

Table 2 - Functions of Environment Sensing Subsystem's AI Modules

AIM	Function
Spatial Attitude Generation	Computes the CAV Spatial Attitude from CAV Centre using GNSS Object and MAS's initial Spatial Attitude.
Audio Scene Description	Produces Audio Scene Descriptors and Alert.
Visual Scene Description	Produces Visual Scene Descriptors and Alert.
LiDAR Scene Description	Produces LiDAR Scene Descriptors and Alert.
RADAR Scene Description	Produces RADAR Scene Descriptors and Alert.
Ultrasound Scene Description	Produces Ultrasound Scene Descriptors and Alert.
Offline Map Scene Description	Produces Offline Map Scene Descriptors.
Basic Environment Description	Produces Basic Environment Descriptors.

8.2.5 I/O Data of AI Modules

For each AIM (1st column), Table 3 gives the input (2nd column) and the output data (3rd column) of the Environment Sensing Subsystem. Note that the Basic Environment Descriptors in column 2 refers to previously produced BED.

Table 3 - I/O Data of Environment Sensing Subsystem's AI Modules

AIM	Input	Output
Audio Scene Description	- Audio Object - Spatial Attitude - Other Scene Descriptors - Basic Environment Descriptors	- Alert - Audio Scene Descriptors
Visual Scene Description	- Visual Object - Spatial Attitude - Other Scene Descriptors - Basic Environment Descriptors	- Alert - Visual Scene Descriptors

LiDAR Scene Description	- LiDAR Object - Spatial Attitude - Other Scene Descriptors - Basic Environment Descriptors	- Alert - LiDAR Scene Descriptors
RADAR Scene Description	- RADAR Object - Spatial Attitude - Basic Environment Descriptors	- Alert - RADAR Scene Descriptors
Spatial Attitude Generation	- GNSS Object - MAS's Spatial Attitude	- Spatial Attitude
Ultrasound Scene Description	- Ultrasound Object - Spatial Attitude - Other Scene Descriptors - Basic Environment Descriptors	- Alert - Ultrasound Scene Descriptors
Offline Map Scene Description	- Offline Map Object - Spatial Attitude	- Offline Map Scene Descriptors
Basic Environment Description	- Audio Scene Descriptors - LiDAR Scene Descriptors - Offline Map Scene Descriptors - RADAR Scene Descriptors - Spatial Attitude - Ultrasound Scene Descriptors - Visual Scene Descriptors - Weather Data - Full Environment Descriptors	- Basic Environment Descriptors

8.2.6 AIW, AIMs, and JSON

Table 4 - AIW, AIMs, and JSON Metadata

AIW	AIM	Name	JSON
CAV-ESS		Environment Sensing Subsystem	X
	OSD-ASD	Audio Scene Description	X
	CAV-BED	Basic Environment Description	X
	CAV-LSD	LiDAR Scene Description	X
	CAV-OSD	Offline Map Scene Description	X
	CAV-RSD	RADAR Scene Description	X
	CAV-SAG	Spatial Attitude Generation	X
	CAV-USD	Ultrasound Scene Description	X
	OSD-VSD	Visual Scene Description	

8.3 Autonomous Mo0tion Subsystem

8.3.1 Functions

The Autonomous Motion Subsystem (AMS):

1. Receives requests to reach a destination from the Human-CAV Interaction Subsystem (HCI).
2. Requests current Position to Environment Sensing Subsystem (ESS).
3. Converses with HCI (and HCI with humans) and settles on a final Route.
4. Makes requests of Full Environment Descriptors subsets to selected CAVs in range.
5. Produces its own Full Environment Descriptors.
6. Receives and responds to requests of Full Environment Descriptors subsets from CAVs in range.
7. Issues Message to Motion Actuation Subsystem (MAS).
8. Processes Message from Motion Actuation Subsystem.
9. Stores Data Receives/Produced in AMS Memory for future use by AMS AIMs.

8.3.2 Reference Model

Figure 1 gives the Autonomous Motion Subsystem Reference Model.

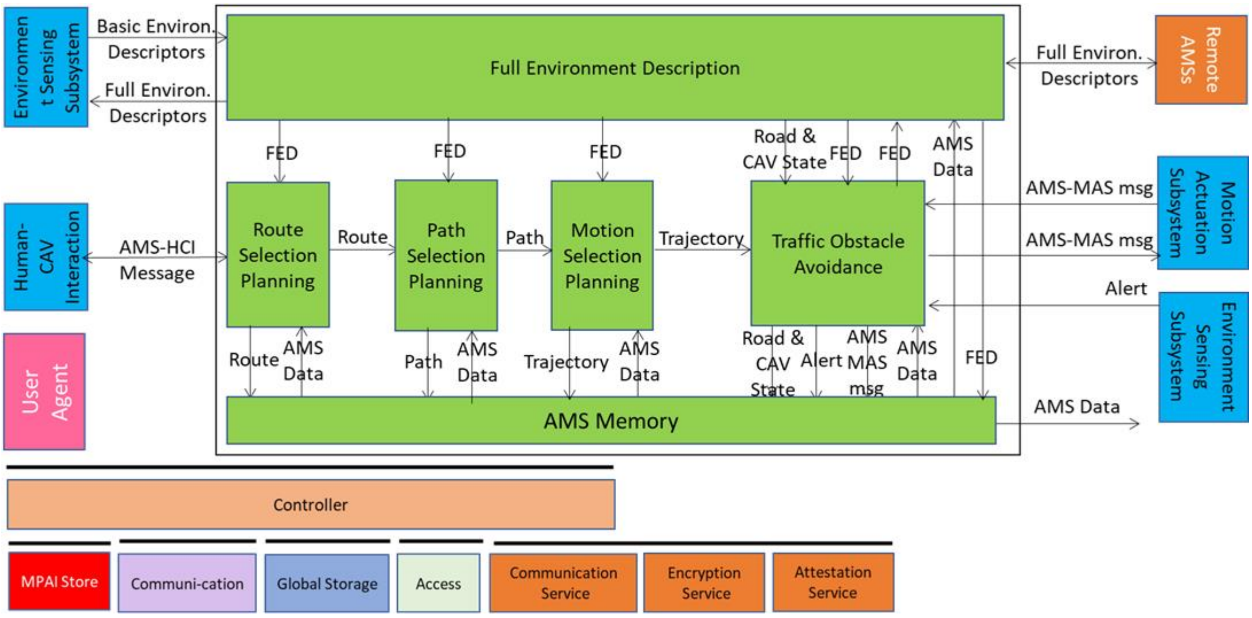

Figure 1 – Autonomous Motion Subsystem Reference Model

The operation of the Autonomous Motion Subsystem unfolds as follows:

1. A human requests the Human-CAV Interaction to take them to a destination.
2. HCI interprets request and passes the interpretation to the AMS.
3. The AMS activates Route Planning to generate a set of Waypoints starting from the current Pose (obtained from the ESS) up to destination.
4. The AMS
    1. Receives Basic Scene Descriptors from the ESS.
    2. Requests (subsets of) Remote AMSs' Full Scene Descriptors and responds to similar requests from Remote AMSs.
    3. Integrates all sources of Environment Descriptors into Full Environment Descriptors
5. The Route's Waypoints cause the Path Selection Planning to generate a set of Positions to reach the next Waypoint.
6. Motion Selection Planning generates a Trajectory to reach the next Position in each Path.

7. Traffic Obstacle Avoidance receives the Trajectory and checks if an Alert was received.
8. If an Alert was received, Traffic Obstacle Avoidance detects whether the Trajectory creates a collision.
    1. If a collision is detected, Traffic Obstacle Avoidance requests a new Trajectory from Motion Planner.
    2. If no collision is detected, Traffic Obstacle Avoidance issues an AMS-MAS Message to MAS.
9. The MAS sends an AMS-MAS Message to AMS informing about the execution of the Command.
10. The AMS, based on the received MAS-AMS Messages, may
    1. Discontinue the execution of the earlier AMS-MAS Message.
    2. Issue a new AMS-MAS Message.
    3. Inform Obstacle Avoidance and Full Environment Description.
11. The decision of each element of the chain may be recorded in the AMS Memory ("black box").

The Trajectory Planning and Decision (CAV-TPD) is a Composite AIM that includes the Path Selection Planning, Motion Selection Planning, and the Traffic Obstacle Avoidance AIMs

8.3.3 I/O Data

Table 1 gives the input/output data of Autonomous Motion Subsystem.

Table 1 - I/O data of Autonomous Motion Subsystem

Input data	From	Comment
Basic Environment Descriptors	Environment Sensing Subsystem	CAV's Environment representation from ESS.
Alert	Environment Sensing Subsystem	Critical information from an EST in ESS.
AMS-HCI Message	Human-CAV Interaction	Human commands, e.g., "take me home".
Full Environment Descriptors	Remote AMSs	Other CAVs and vehicles, and roadside units.
AMS-MAS Message	Motion Actuation Subsystem	Message sent by the AMS to the MAS.
Ego-Remote AMS Message	Remote AMS	Remote AMS to Ego AMS message.
Output data	**To**	**Comment**
AMS-HCI Message	Human-CAV Interaction	AMS's message to HCI-AMS.
AMS-MAS Message	Motion Actuation Subsystem	Message to MAS, e.g., "in 5s assume a given Spatial Attitude".
Full Environment Descriptors	Remote AMSs	To Ego CAV's ESS and to REmote CAVs.
Ego-Remote AMS Message	Remote AMSs	Ego AMS to Remote AMS message.
AMS Data	External Device	For offline analysis.

8.3.4 Functions of AI Modules

Table 2 gives the AI Modules of the Autonomous Motion Subsystem.

Table 2 - Functions of Autonomous Motion Subsystem's AI Modules

AIM	Function
Full Environment Description	Creates an internal environment representation by fusing information received from ESS, Remote AMSs, and other CAV-aware entities. Updates the CAV State.
Route Selection Planning	Computes a set of possible Routes, through the road network, from the current to the target destination.
Path Selection Planning	Generates a set of Paths, considering: 1. Route. 2. Full Environment Descriptors (Spatial Attitude, Road State, etc.). 4. Traffic Rules.
Motion Selection Planning	Defines a Trajectory to reach a Goal using the Spatial Attitude considering: 1. CAV's kinematic and dynamic constraints. 2. Full Environment Descriptors 3. Passengers' comfort.
Traffic Obstacle Avoidance	Checks whether Trajectory is compatible with Alert information: if it is not, it requests a new Trajectory; if it is, it instructs the MAS to execute the Trajectory considering the Environment conditions and receives MAS-AMS Messages about the execution. Based on a Message, updated Road State and CAV State may be communicated to Obstacle Avoidance.
AMS Memory	Records decisions by Route Planning, Path Planning, Motion Planning, Obstacle Avoidance, Full Environment Description, and Command Issuance.

9 5 I/O Data of AI Modules

Table 3 gives, for each AIM (1st column), the input data (2nd column) and the output data (3rd column) of Autonomous Motion Subsystem.

Table 3 - Autonomous Motion Subsystem's data

AIM	Input	Output
Full Environment Description	- Basic Environment Descriptors - Full Environment Descriptors - AMS Data - Road State - CAV State	- Full Environment Descriptors
Route Selection Planning	- Full Environment Descriptors - AMS Data - AMS-HCI Message - Selected Route - Route ID	- AMS-HCI Message - Route
Path Selection Planning	- Full Environment Descriptors - AMS Data - Route	- Paths
Motion Selection Planning	- Full Environment Descriptors - AMS Data - Paths	- Trajectory
Traffic Obstacle Avoidance	- Full Environment Descriptors - Trajectory - AMS Data	- Full Environment Descriptors - AMS-MAS Message - Road State

	- Alert - AMS-MAS Message	- CAV State - Alert
AMS Memory	- Full Environment Descriptors - Route - Path - Trajectory - Alert - Road State - CAV State - AMS-MAS Message	- AMS Data

9.1.1 AIW, AIMs, and JSON Metadata

AIW	AIMs	Name	JSON
CAV-AMS		Autonomous Motion Subsystem	X
	CAV-FEV	Full Environment Description	X
	CAV-RSP	Route Selection Planning	X
	CAV-PSP	Path Selection Planning	X
	CAV-MSP	Motion Selection Planning	X
	CAV-TOA	Traffic Obstacle Avoidance	X
	CAV-AMM	AMS Memory	

9.2 Motion Actuation Subsystem

9.2.1 Functions of Motion Actuation Subsystem

The Motion Actuation Subsystem (MAS):

1. Transmits spatial and weather information gathered from its sensors and mechanical subsystems to the Environment Sensing Subsystem (ESS).
2. Receives AMS-MAS Messages from the Autonomous Motion Subsystem (AMS).
3. Translates AMS-MAS Message into specific Commands to its own Brake, Motor, and Wheel mechanical subsystems.
4. Receives Responses from its Brake, Motor, and Wheel mechanical subsystems.
5. Packages Responses into and sends AMS-MAS Messages to Autonomous Motion Subsystem.

9.2.2 Reference Architecture of Motion Actuation Subsystem

Figure 1 represents the Reference Model of the Motion Actuation Subsystem (CAV-MAS).

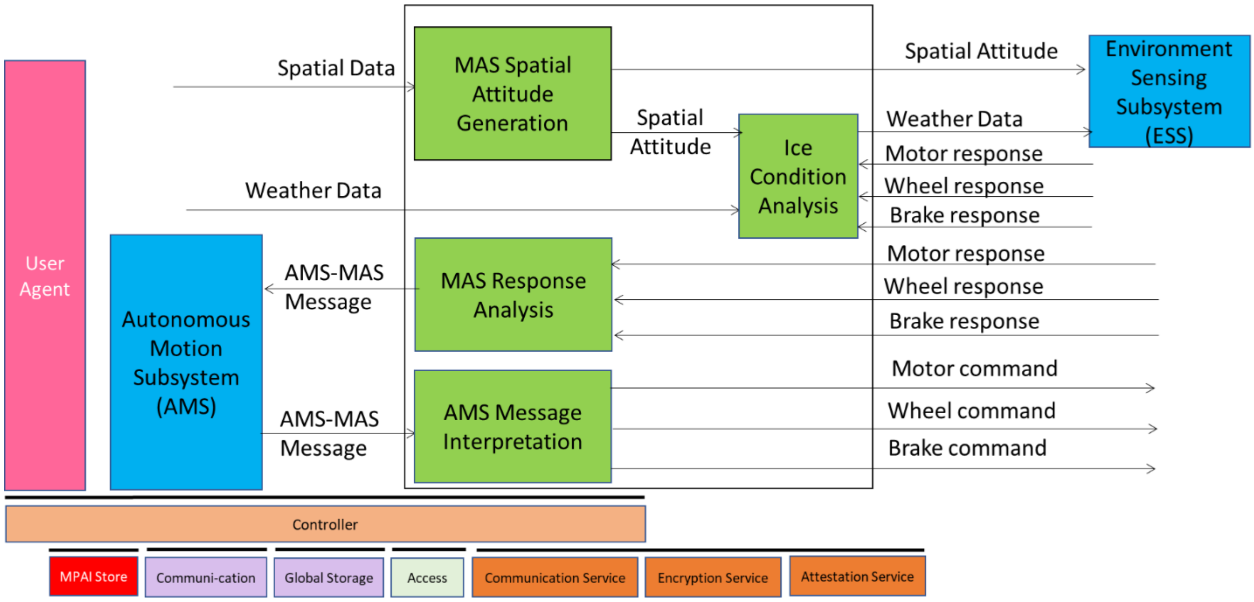

Figure 1 – Motion Actuation Subsystem Reference Model

The operation of the Motion Actuation Subsystem unfolds as follows:

1. AMS Command Interpretation
   1. Interprets the AMS-MAS Messages received from AMS and issues commands to the Brake, Motor, and Wheel mechanical subsystems.
2. MAS Response Analysis
   1. Interprets the responses received from the Brake, Motor, and Wheel mechanical subsystems and sends AMS-MAS Messages to to AMS.
3. MAS Spatial Attitude Generation
   1. Computes the initial Ego CAV's Spatial Attitude Attitude using Spatial Data (Odometer, Speedometer, Accelerometer, and Inclinometer) Data
   2. Sends the initial Spatial Attitude Attitude to the ESS.
4. Ice Condition Analysis
   1. Augments Weather Data analysing the responses of the Brake, Motor, and Wheel mechanical subsystems.
   2. Sends augmented Weather Data to ESS.

9.2.3 I/O Data of Motion Actuation Subsystem

Table 1 gives the input/output data of Motion Actuation Subsystem.

Table 1 - I/O data of Motion Actuation Subsystem

Input	Comments
Spatial Data	Collection of distance, velocity, acceleration, and inclination data.
Weather Data	Data such as humidity, pressure, temperature.
AMS-MAS Message	Message including motion information.
Motor Response	Information on effects of applied motor force.
Wheel Response	Information on effects of applied Wheel rotation force.
Brake Response	Information on effects of applied brake force.
Output	**Comments**
Spatial Attitude	Position, Orientation and their velocity and acceleration vectors.
Weather Data	Data such as humidity, pressure, temperature, ice condition.

Motor Command	Applied motor torque.
Wheel Command	Applied wheel torque.
Brake Command	Applied brake deceleration.
AMS-MAS Message	Message including results of MAS Response analysis.

9.2.4 Functions of Motion Actuation Subsystem's AI Modules

Table 2 gives the AI Modules of Autonomous Motion Subsystem.

Table 2 - Functions of Motion Actuation Subsystem's AI Modules

AIM	Function
MAS Spatial Attitude Generation	Computes Ego CAV's Spatial Attitude using Spatial Data.
AMS Message Interpretation	Receives, analyses, and actuates AMS-MAS Message into specific commands to Brakes, Wheels, and Motors.
MAS Response Analysis	Receives and analyses responses from Brakes, Wheel, and Motors and sends the MAS-AMS Response to AMS.
Ice Condition Analysis	Adds ice condition information to input Weather Data.

9.2.5 I/O Data of Motion Actuation Subsystem's AI Modules

Table 3 gives, for each AIM (1st column), the input data (2nd column) from which AIM (column) and the output data (3rd column).

Table 3 - I/O Data of Motion Actuation Subsystem's AI Modules

AIM	Input	Output
MAS Spatial Attitude Generation	- Spatial Data	- Spatial Attitude
AMS Command Interpretation	- AMS-MAS Message	- Brake Command - Motor Command - Wheel Command
MAS Response Analysis	- Brake Response - Motor Response - Wheel Response	- AMS-MAS Message
Ice Condition Analysis	- Brake Response - Motor Response - Wheel Response - Weather Data	- Weather Data

9.2.6 AIW, AIMs, and JSON

AIW	AIMs	AIM Names	JSON
CAV-MAS		Motion Actuation Subsystem	X
	CAV-MSG	MAS Spatial Attitude Generation	X
	CAV-AMI	AMS-MAS Message Interpretation	X
	CAV-MRA	MAS Response Analysis	X
	CAV-ICA	Ice Condition Analysis	X

10 AI Modules

Table 1 provides the links to the AI Modules part of the four CAV Subsystem. Note that the Human-CAV Interaction Subsystem is specified by Multimodal Conversation (MPAI-MMC).

Table 1 - AI Modules used by MPAI-CAV organised by AI Workflows

Human-CAV Interaction	Environment Sensing	Autonomous Motion	Motion Actuation
Audio-Visual Scene Description	Audio Scene Description	AMS Memory	AMS-MAS Message Interpretation
Automatic Speech Recognition	Basic Environment Description	Full Environment Description	Ice Condition Analysis
Audio-Visual Alignment	LiDAR Scene Description	Motion Selection Planning	MAS Response Analysis
Audio-Visual Scene Rendering	Offline Map Scene Description	Path Selection Planning	MAS Spatial Attitude Generation
Natural Language Understanding	RADAR Scene Description	Route Selection Planning	
Entity Dialogue Processing	Spatial Attitude Generation	Traffic Obstacle Avoidance	
Face Identity Recognition	Ultrasound Scene Description	Trajectory Planning and Decision	
Personal Status Display	Visual Scene Description		
Personal Status Extraction			
Speaker Identity Recognition			
Visual Object Identification			

AIMs are sequentially specified in by Subsystems.

10.1 Audio-Visual Scene Description

10.1.1 Functions

Audio-Visual Scene Description (OSD-AVS) produces Audio-Visual Scene Descriptors from Speech, Audio, Visual and Audio-Visual Scene Descriptors:

Receives	*Space-Time*	Of output Audio-Visual Scene Descriptors,
	Speech Objects	

Audio Objects

Visual Objects

Audio-Visual Scene Descriptors Of Scene to be augmented.

Augments *Audio-Visual Scene Descriptors*

Produces *Audio-Visual Scene Descriptors*

10.1.2 Reference Model

Figure 1 specified the Reference Model of Audio-Visual Scene Description (OSD-AVS) aim.

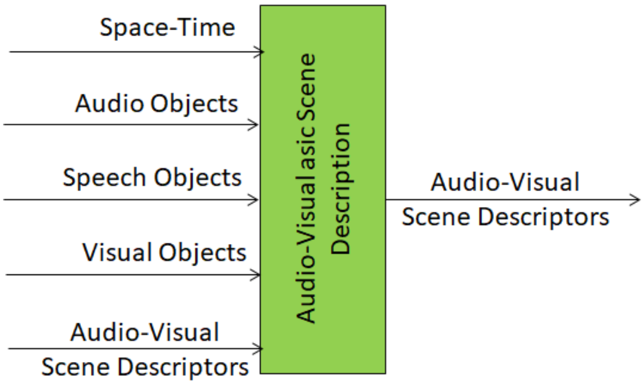

Figure 1 – The Audio-Visual Scene Description (OSD-AVS) AIM

10.1.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Audio-Visual Scene Description (OSD-AVS) AIM. Links are to the Data Type specifications.

Table 1 – I/O Data of the Audio-Visual Scene Description (OSD-AVS) AIM

Input	Description
Space-Time	Space-Time information of output Audio-Visual Scene Descriptors
Speech Object	Speech Object
Audio Objects	Audio Objects.
Visual Objects	Visual Objects.
Audio-Visual Scene Descriptors	The Audio-Visual Descriptors of the Scene part of the target Audio-Visual Scene.
Output	**Description**
Audio-Visual Scene Descriptors	The Audio-Visual Descriptors of the Scene.

10.1.4 SubAIMs

Figure 2 specified the Reference Model of Audio-Visual Scene Description (CAE-ASD) Composite AIM.

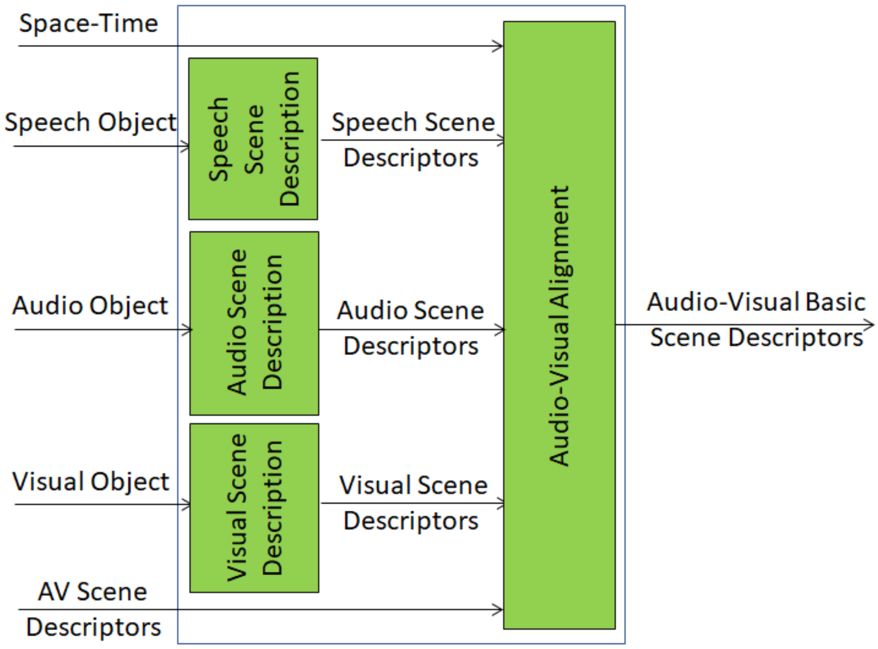

Figure 2 – The Audio-Visual Scene Description (OSD-AVS) Composite AIM

Table 2 provides the links to the specifications of the OSD-AVS AIMs.

Table 2 – AIMs of the Audio-Visual Scene Description (OSD-AVS) Composite AIM

AIMs	Names	JSON
MMC-SSD	Speech Scene Description	X
CAE-ASD	Audio Scene Description	X
OSD-VSD	Visual Scene Description	X
OSD-AVA	Audio-Visual Alignment	X

10.1.5 JSON Metadata

https://schemas.mpai.community/OSDV1.3/AIMs/AudioVisualSceneDescription.json

10.1.6 Reference Software

10.1.6.1 Disclaimers

1. This OSD-AVS Reference Software Implementation is released with the BSD-3-Clause licence.
2. The purpose of this OSD-AVS Reference Software is to show a working Implementation of OSD-AVS, not to provide a ready-to-use product.
3. MPAI disclaims the suitability of the Software for any other purposes and does not guarantee that it is secure.
4. Use of this Reference Software may require acceptance of licences from the respective repositories. Users shall verify that they have the right to use any third-party software required by this Reference Software.

10.1.6.2 Guide to the OSD-AVS code

OSD-AVS arranges the aligned visual and speech objects into Audio-Visual Scene Descriptors.

Use of this Reference Software for the OSD-AVS AI Module is for developers who are familiar with Python, Docker, and RabbitMQ.

The OSD-AVS Reference Software is found at the MPAI gitlab site. It contains:

1. src: a folder with the Python code implementing the AIM
2. Dockerfile: a Docker file containing only the libraries required to build the Docker image and run the container
3. requirements.txt: dependencies installed in the Docker image.

10.1.6.3 Acknowledgements

This OSD-AVS Reference Software has been developed by the MPAI *AI Framework* Development Committee (AIF-DC).

10.1.7 Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-AVS AIM. AIM. Conformance Testing of the individual AIMs of the OSD-AVS Composite AIM are given by the individual AIM Specification.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-AVS AIM

Receives	Space-Time	Shall validate against Space-Time schema.
	Speech Objects	Shall validate against Speech Objects schema. Speech Data shall conform with Qualifier.
	Audio Objects	Shall validate against Audio Objects schema. Audio Data shall conform with Qualifier.
	Visual Objects	Shall validate against Visual Objects schema. Visual Data shall conform with Qualifier.
Produces	Audio-Visual Scene Descriptors	Shall validate against AV Scene Descriptors schema.

10.2 Automatic Speech Recognition

10.2.1 Functions

Automatic Speech Recognition (MMC-ASR):

Receives	Language Selector	Signalling the language of the speech.
	Auxiliary Text	Text that may be used to provide context information.

	Speech Object	Speech to be recognised.
	Speaker ID	ID of speaker uttering speech.
	Speech Overlap	Data type providing information of speech overlap.
	Speaker Time	Time during which the speech is to be recognised.
Produces	Recognised Text	(Also called text transcript).

Recognised Text can be a Text Segment or just a string.

10.2.2 Reference Model

Figure 1 depicts the Reference Model of the Automatic Speech Recognition (MMC-ASR) AIM.

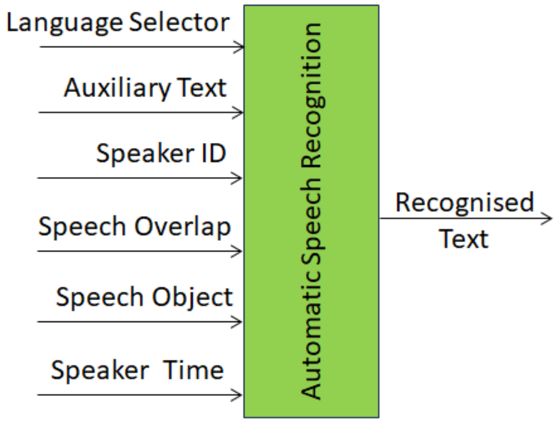

Figure 1 – The Automatic Speech Recognition (MMC-ASR) AIM

10.2.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Automatic Speech Recognition (MMC-ASR) AIM.

Table 1 – I/O Data of the Automatic Speech Recognition (MMC-ASR) AIM

Input	Description
Language Selector	Selects input language
Auxiliary Text	Text Object with content related to Speech Object.
Speech Object	Speech Object emitted by Entity
Speaker ID	Identity of Speaker
Speech Overlap	Times and IDs of overlapping speech segments
Speaker Time	Time during which Speech is recognised
Output	**Description**
Recognised Text	Output of the Automatic Speech Recognition AIM, a Text Segment or just a string.

10.2.4 JSON Metadata

https://schemas.mpai.community/MMC/V2.3/AIMs/AutomaticSpeechRecognition.json

10.2.5 Reference Software

10.2.5.1 Disclaimers

1. This MMM-ASR Reference Software Implementation is released with the BSD-3-Clause licence.
2. The purpose of this Reference Software is to demonstrate a working Implementation of MMC-ASR, not to provide a ready-to-use product.
3. MPAI disclaims the suitability of the Software for any other purposes and does not guarantee that it is secure.
4. Use of this Reference Software may require acceptance of licences from the respective repositories. Users shall verify that they have the right to use any third-party software required by this Reference Software.

10.2.5.2 Guide to the ASR code #1

The code takes Speech Objects from MMC-AUS and generates Text Segments (called text transcripts). It uses the whisper-large-v3 model to convert an input Speech Object (speaker's turn) into a Text Segment (here called text transcript). Disfluencies (e.g., repetitions, repairs, filled pauses) are often omitted. The Whisper reference document is available.

The MMC-ASR Reference Software is found at the MPAI gitlab site. Use of this AI Modules is for developers who are familiar with Python, Docker, RabbitMQ, and downloading models from HuggingFace. The Reference Software contains:

1. src: a folder with the Python code implementing the AIM
2. Dockerfile: a Docker file containing only the libraries required to build the Docker image and run the container
3. requirements.txt: dependencies installed in the Docker image
4. README.md: commands for cloning https://huggingface.co/openai/whisper-large-v3

Library: https://github.com/linto-ai/whisper-timestamped

10.2.5.3 Guide to the ASR code #2

Use of this AI Modules is for developers who are familiar with Python and downloading models from HuggingFace,

A wrapper for the Whisper NN Module:

1. Manages input files and parameters: Speech Object
2. Performs Speech Recognition on each Speech Object by executing the Whisper Module.
3. Outputs Recognised Text.

The MMC-ASR Reference Software is found at the NNW gitlab site (registration required). It contains:

1. The python code implementing the AIM.
2. The required libraries are: pytorch and transformers (HuggingFace).

10.2.5.4 Acknowledgements

This version of the MMC-ASR Reference Software

1. #1 has been developed by the MPAI *AI Framework* Development Committee (AIF-DC).
2. #2 has been developed by the MPAI *Neural Network Watermarking* Development Committee (NNW-DC).

10.2.6 Conformance Testing

Table 2 provides the Conformance Testing Method for MMC-ASR AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 - MMC-ASR AIM Conformance Testing

Input	Language Selector	Shall validate against the Language Selector part of the schema.
	Auxiliary Text	Shall validate against the Text Object schema. Text Data shall conform with the Text Qualifier.
	Speech Object	Shall validate against the Speech Object schema. Speech Data shall conform with the Speech Qualifier.
	Speaker ID	Shall validate against the Instance ID schema.
	Speech Overlap	Shall validate against the Speech Overlap schema.
	Speaker Time	Shall validate against the Time schema.
Output	Text Object	Shall validate against the Text Object schema. Text Data shall conform with the Text Qualifier, e.g. output language shall be that indicated by the Language Selector,

Table 3 provides an example of MMC-ASR AIM Conformance Testing.

Table 3 - An example of MMC-ASR AIM Conformance Testing

Input Data	**Data Format**	**Input Conformance Testing Data**
Speech Object	.wav	All input Speech files to be drawn from Speech files.
Output Data	**Data Format**	**Output Conformance Testing Criteria**
Recognised Text	Unicode	All Text files produced shall conform with Text files.

10.2.7 Performance Assessment

Performance Assessment of an ASR Implementation (ASRI) can be performed for a language for which there is a dataset of speech segments of various durations with corresponding Transcription Text. An MMC-ASR AIM Performance Assessment Report shall be based on the following steps and specify the input dataset used.

For each Recognised Text produced by the ASRI being Assessed for Performance in response to a speech segment provided as input:

1. Compare the Recognised Text with the Transcription Text
2. Compute the Word Error Rate (WER) defined as the sum of deletion, insertion, and substitution errors in the Recognised Text compared to the Transcription Text, divided by the total number of words in the Transcription Text.

This code can be used to compute the WER.

Performance Assessment of an ASRI for a language in a Performance Assessment Report is defined as "The WER computed on all speech segments included in the reported dataset".

10.3 Audio-Visual Alignment

10.3.1 Functions

Audio-Visual Alignment (OSD-AVA) V1.3 provides the Descriptors of an Audio-Visual Scene whose Audio and Visual Objects that have the same Position, have compatible Identifiers.

Receives	*Speech Scene Descriptors*	Descriptors of potentially present Speech Scene.
	Audio Scene Descriptors	Descriptors of potentially present Audio Scene.
	Visual Scene Descriptors	Descriptors of Visual Scene.
Aligns	Speech, Audio, and Visual Objects	Sharing the same Spatial Attitude
Produces	*Audio-Visual Scene Descriptors*	Where Speech Objects, Audio Objects and Visual Objects having the same Spatial Attitude have compatible Identifiers.

10.3.2 Reference Model

Figure 1 specifies the Reference Model of the Audio-Visual Alignment (OSD-AVA) AIM.

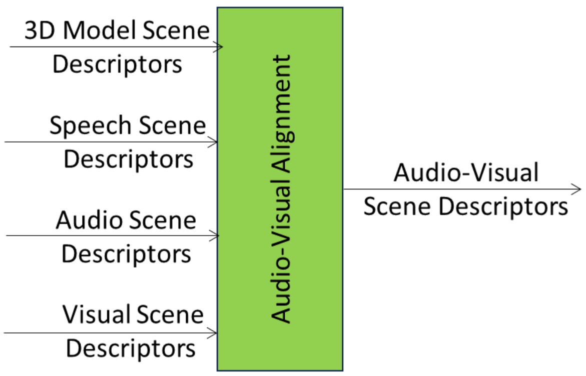

Figure 1 - Reference Model of the Audio-Visual Alignment (OSD-AVA) AIM

10.3.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Audio-Visual Alignment (OSD-AVA) AIM.

Table 1 – I/O Data of the Audio-Visual Alignment AIM

Input	Description
Speech Scene Descriptors	The IDs and the geometry of the Speech Objects of the Scene.
Audio Scene Descriptors	The IDs and the geometry of the Audio Objects of the Scene.
Visual Scene Descriptors	The IDs and the geometry of the Audio Objects of the Scene.
Output	**Description**
Audio-Visual Scene Descriptors	The IDs and the geometry of the Audio, Visual and Audio-Visual Objects of the Scene.

10.3.4 JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/AudioVisualAlignment.json

10.3.5 Reference Software

10.3.5.1 Disclaimers

1. This OSD-AVA Reference Software Implementation is released with the BSD-3-Clause licence.
2. The purpose of this Reference Software is to show a working Implementation of OSD-AVA, not to provide a ready-to-use product.
3. MPAI disclaims the suitability of this Reference Software for any other purposes and does not guarantee that it is secure.
4. Use of this Reference Software may require acceptance of licences from the respective repositories. Users shall verify that they have the right to use any third-party software required by this Reference Software.

10.3.5.2 Guide to OSD-AVA code

OSD-AVA arranges the output Visual Objects and Speech Objects with corresponding Time information: scene cuts/transitions and speakers' turns. Each Object is bounded by two adjacent times from a list of unique times that are either 1) scene cuts/transitions or 2) starts and ends of speakers' turns.

Use of this Reference Software for the OSD-AVA AI Module is for developers who are familiar with Python, Docker, and RabbitMQ.

OSD-AVA computes segments as unique intervals from scene bounds and from speech segments. Moreover, OSD-AVA outputs visual objects and speech objects.

The OSD-AVA Reference Software is found at the MPAI gitlab site. It contains:

1. src: a folder with the Python code implementing the AIM
2. Dockerfile: a Docker file containing only the libraries required to build the Docker image and run the container
3. requirements.txt: dependencies installed in the Docker image.

10.3.5.3 Acknowledgements

This version of the MMC-ASR Reference Software has been developed by the MPAI *AI Framework* Development Committee (AIF-DC).

10.3.6 Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-AVA AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-AVA AIM

Receives	Speech Scene Descriptors	Shall validate against Speech Scene Descriptors schema
	Audio Scene Descriptors	Shall validate against Audio Scene Descriptors schema
	Visual Scene Descriptors	Shall validate against Visual Scene Descriptors schema
Produces	Audio-Visual Scene Descriptors	Shall validate against AV Scene Descriptors schema

10.3.7 Performance Assessment

Performance Assessment of an OSD-AVA AIM Implementation shall be performed using a dataset of scenes containing Audio and/or Speech and Visual objects.

The Performance Assessment Report of an OSD-AVA AIM Implementation shall include:

1. The Identifier of the OSD-AVA AIM whose Performance is being Assessed.
2. The Identifier of the scene dataset used which include the identifiers of the aligned objects.
3. The data type of the scenes: analogue, digital, without or with separated objects.
4. The Performance of the OSD-AVA AIM expressed as:
    1. The number of times the OSD-AVA AIM being Assessed for Performance correctly identifies as aligned the objects that the data set declares as aligned divided by the total number of aligned objects (Truly aligned objects).
    2. The number of time the OSD-AVA AIM being Assessed for Performance incorrectly identifies as aligned the object that the dataset declares aligned in the dataset divided by the total number of aligned objects (Falsely aligned objects).
    3. The number of time the OSD-AVA AIM being Assessed for Performance incorrectly identifies as non-aligned object that are declared aligned in the dataset referenced in 2 divided by the total number of aligned objects (Missed aligned objects).

10.4 Audio-Visual Scene Rendering

10.4.1 Functions

Audio-Visual Scene Rendering (PAF-AVR) produces Speech, Audio, and Visual Objects from a Portable Avatar, Audio-Visual Scene Descriptors and a Point of View:

Receives	Point of View	To be used in rendering the scene and its objects.
	Audio-Visual Scene Descriptors	jointly with or alternatively with Portable Avatar.
	Portable Avatar	Jointly with or alternatively with AV Scene Descriptors.
Transforms	Portable Avatar	Into generic Audio-Visual Scene Descriptors if input is Portable Avatar.
Produces	Portable Avatar's Output Speech	Always integrated in the Audio-Visual Scene. Output Speech results from the rendering of Audio Scene Descriptors from human-selected Point of View.
	Output Audio	Resulting from the rendering of Audio Scene Descriptors from human-selected Point of View.
	Output Visual	Resulting from the rendering of Audio Scene Descriptors from human-selected Point of View. View Selector tells the OSD-AVR AIM where the visual components of the Portable Avatar should also be integrated.

10.4.2 Reference Model

Figure 1 specifies the Reference Model of the Audio-Visual Scene Rendering (PAF-AVR) AIM.

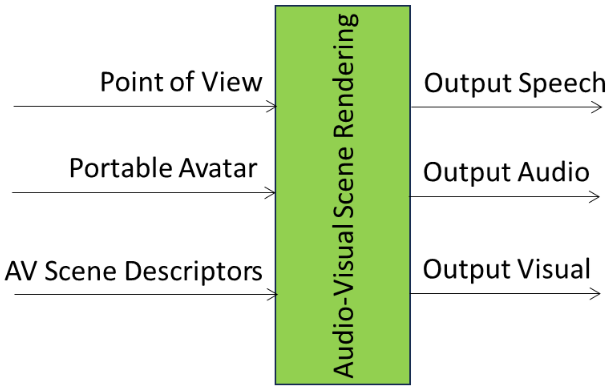

Figure 1 – The Audio-Visual Scene Rendering (PAF-AVR) AIM

10.4.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Audio-Visual Scene Rendering (PAF-AVR) AIM.

Table 1 – I/O Data of the Audio-Visual Scene Rendering (PAF-AVR) AIM

Input	Description
Portable Avatar	Data produced, e.g., by Personal Status Display.
AV Scene Descriptors	Audio-Visual Scene Descriptors.
Point of View	Point from where an Entity perceives the Audio-Visual Scene
Output	**Description**

Output Speech Object	The Speech components of the Audio-Visual Scene.
Output Audio Object	The Audio components of the Audio-Visual Scene.
Output Visual Object	The Visual components of the Audio-Visual Scene.

10.4.4 JSON Metadata

https://schemas.mpai.community/PAF/V1.4/AIMs/AudioVisualSceneRendering.json

10.4.5 Profiles

The Profiles of Audio-Visual Scene Rendering are specified.

10.4.6 Conformance Testing

Table 2 provides the Conformance Testing Method for PAF-AVR AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for PAF-AVR AIM

Receives	Portable Avatar	Shall validate against Point of View Schema.
	AV Scene Descriptors	Shall validate against AV Scene Descriptors Schema.
	Point of View	Shall validate against Portable Avatar Schema. Portable Avatar Data shall conform with respective Qualifiers.
Produces	Output Speech Object	Shall validate against Speech Object Schema. Speech Data shall conform with Speech Qualifier.
	Output Audio Object	Shall validate against Audio Object Schema. Audio Data shall conform with Audio Qualifier.
	Output Visual Object	Shall validate against Visual Object or 3D Model Schema. Visual Data shall conform with Visual Object.

10.5 Natural Language Understanding

10.5.1 Functions

Natural Language Understanding (MMC-NLU):

Receives	Text Object directly input by the Entity. Recognised Text from an Automatic Speech Recognition AIM. The ID of an Instance. The Audio-Visual Scene Descriptors containing the Instance ID.
Refines	Input Text if coming from an Automatic Speech Recognition AIM

Extracts	Meaning (Text Descriptors) from Recognised Text or Entity's Text Object.
Produces	Refined Text.
	Text Descriptors (Meaning).
Enables	Personal Stats Display to produce a Portable Avatar.

10.5.2 Reference Model

Figure 1 specifies the Reference Model of the Natural Language Understanding (MMC-NLU) AIM.

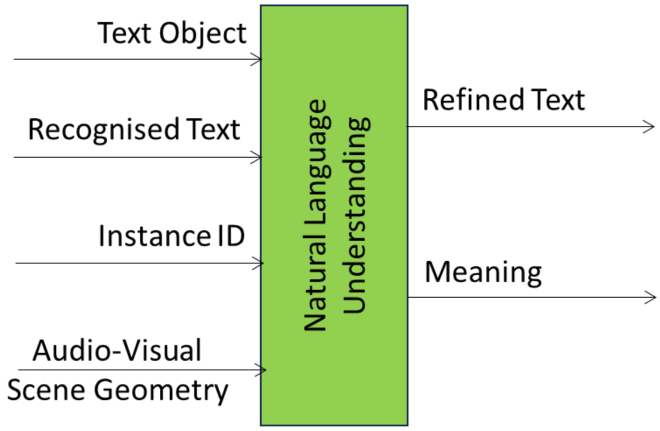

Figure 1 – The Natural Language Understanding (MMC-NLU) AIM Reference Model

10.5.3 3 Input/Output Data

Table 1 specifies the Input and Output Data of the Natural Language Understanding (MMC-NLU) AIM.

Table 1 – I/O Data of the Natural Language Understanding (MMC-NLU) AIM

Input	Description
Text Object	Input Text.
Recognised Text	Text from the Automatic Speech Recognition AIM.
Instance ID	The Identifier of the specific Audio or Visual Object belonging to a level in the taxonomy.
Audio-Visual Scene Geometry	The digital representation of the spatial arrangement of the Visual Objects of the Scene.
Visual Instance ID	The Identifier of the specific Visual Object belonging to a level in the taxonomy.
Output	**Description**
Meaning	Descriptors of the Refined Text.
Refined Text	The refined version of the Recognised Text from the NLU AIM.

10.5.4 JSON Metadata

https://schemas.mpai.community/MMC/V2.3/AIMs/NaturalLanguageUnderstanding.json

10.5.5 Profiles

The Profiles of the Natural Language Understanding (MMC-NLU) AIM are specified.

10.5.6 Conformance Testing

Table 2 provides the Conformance Testing Method for MMC-NLU AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for MMC-NLU AIM

Input	Text Object	Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
	Recognised Text	Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
	Instance ID	Shall validate against Instance ID schema.
	Audio-Visual Scene Geometry	Shall validate against AV Scene Descriptors schema.
Output	Refined Text	Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
	Meaning	Shall validate against Meaning schema.

Table 3 provides an example of MMC-NLU AIM conformance testing.

Table 3 – An example MMC-NLU AIM conformance testing

Input Data	**Data Type**	**Input Conformance Testing Data**
Input Selector	Binary data	All Input Selectors shall conform with Selector.
Text Object	Unicode	All input Text files to be drawn from Text files.
Recognised Text	Unicode	All input Text files to be drawn from Text files.
Output Data	**Data Type**	**Output Conformance Testing Criteria**
Meaning	JSON	All JSON files shall validate against Meaning Schema
Refined Text	Unicode	All Text files produced shall conform with Text.

The four taggings: POS_tagging, NE_tagging, dependency_tagging, and SRL_tagging must be present in the output JSON file of Meaning. Any of the four tagging values may be null.

10.6 Entity Dialogue Processing

10.6.1 Functions

Entity Dialogue Processing (MMC-EDP):

Receives	Text Object	Text of the entity upstream to be processed.
	Object Instance ID	Of an object in a scene.
	Personal Status	of the entity upstream.
	Text Descriptors	Descriptors of input Text Object.
	AV Scene Geometry	Geometry of the AV scene containing object whose ID is provided.
	Speaker ID	ID of speaker uttering the speech that contains the Text Object.
	Face ID	ID of the face of the speaker uttering the speech that contains the Text Object.
	Summary	A summary of the discussions being held in the environment.
Handles	One Text Object at a time	From an entity upstream.
Recognises	The identity	Of entity upstream using speech and/or face.
Takes into account	Past Text Objects	and their spatial arrangement.
Produces	Summary	Edited summary based on input data.
	Text Object	of Machine.
	Personal Status	of Machine.

10.6.2 Reference Model

Figure 1 depicts the Reference Model of the Entity Dialogue Processing (MMC-EDP) AIM.

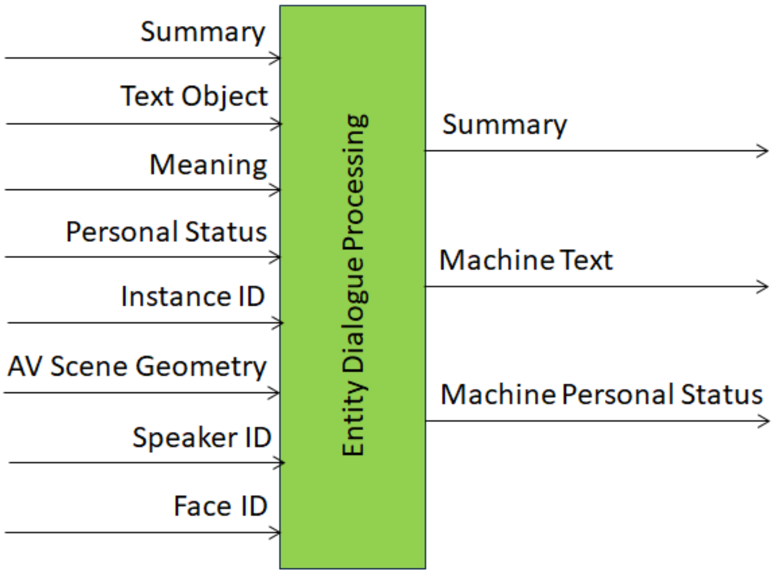

Figure 1 – Entity Dialogue Processing (MMC-EDP) AIM Reference Model

10.6.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Entity Dialogue Processing (MMC-EDP) AIM.

Table 1 – I/O Data of the Entity Dialogue Processing (MMC-EDP) AIM

Input	Description
Summary	The summary in the current state.
Text Object	Text or Refined Text from the Entity the Machine is communicating with.
Meaning	Descriptors of Text and/or Translated Text of the Entity the Machine is communicating with.
Personal Status	Personal Status of the Entity the Machine is communicating with.
Instance ID	ID of the Audio of Visual Object the Entity refers to.
Audio-Visual Scene Geometry	The Geometry of the AV Scene.
Speaker ID	The ID of the Speaker.
Face ID	The ID of the Face.
Output	**Description**
Machine Text	Text produced by the Machine in response to input.
Machine Personal Status	The Personal Status the Machine intends to add to its Modalities.
Summary	The result of refining the input Summary taking comments into consideration.

10.6.4 JSON Metadata

https://schemas.mpai.community/MMC/V2.3/AIMs/EntityDialogueProcessing.json

10.6.5 Profiles

Profiles of Entity Dialogue Processing are specified.

10.6.6 Conformance Testing

Table 2 provides the Conformance Testing Method for MMC-EDP AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – MMC-EDP AIM Conformance Testing

Input	Text Object	Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
	Object Instance ID	Shall validate against Instance Identifier schema.
	Input Personal Status	Shall validate against Personal Status schema.
	Meaning	Shall validate against Text Descriptors schema.
	Audio-Visual Scene Geometry	Shall validate against AV Scene Geometry schema.
	Speaker ID	Shall validate against Instance ID schema.
	Face ID	Shall validate against Face ID schema.
	Summary	Shall validate against Summary schema. Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
Output	Edited Summary	Shall validate against Summary schema. Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
	Machine Text Object	Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
	Machine Personal Status	Shall validate against Personal Status schema.

Table 3 provides an example of MMC-EDP AIM Conformance Testing.

Table 3 – An example of MMC-EDP AIM Conformance Testing

Input Data	**Data Type**	**Input Conformance Testing Data**
Meaning	JSON	All input JSON Emotion files to be drawn from Meaning JSON Files
Recognised Text	Unicode	All input Text files to be drawn from Text files.
Input Emotion	JSON	All input JSON Emotion files to be drawn from Emotion JSON Files
Output Data	**Data Type**	**Output Conformance Testing Criteria**
Machine Text	Unicode	All Text files produced shall conform with Text.
Machine Emotion	JSON	Emotion JSON Files shall validate against Emotion Schema

The two attributes emotion_Name and emotion_SetName must be present in the output JSON file of Emotion. The value of either of the two attributes may be null.

10.7 Face Identity Recognition

10.7.1 Functions

Face Identity Recognition (PAF-FIR) produces the Bounding Box with the face and the identity of the face from the image and the geometry of the Visual Scene the Image Visual Object it belongs to:

Receives	*Text Object*	Text that is related with the Face to be identified.
	Image Visual Object	Image containing Face to be identified.
	Face Time	Time when the face should be identified.
	Visual Scene Geometry	Of the scene where the Face is located.
Searches for	*Bounding Boxes*	That include faces
Finds	best match	Between the Faces and those in a database.
Produces	*Face Identities*	Face Instance Identifiers.
	Bounding Boxes	Bounding Boxes that include faces.

10.7.2 Reference Model

Figure 1 depicts the Reference Model of the Face Identity Recognition AIM.

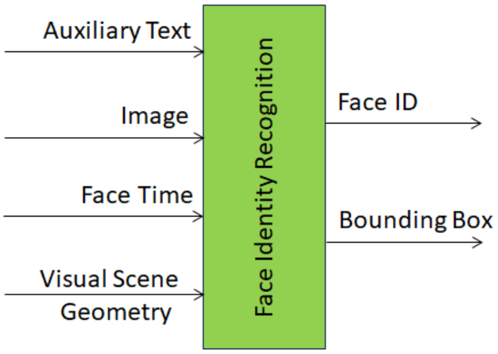

Figure 1 – Face Identity Recognition AIM

10.7.3 Input/Output Data

Table 1 specifies the Input and Output Data of the of the Face Identity Recognition AIM.

Table 1 – I/O Data of the Face Identity Recognition AIM

Input	Description
Auxiliary Text Objext	Text with a content related to Face ID.
Image Visual Object	An image containing the Face to be identified.
Face Time	The Time during which the Face should be identified.
Visual Scene Geometry	The Geometry of the Scene where the Face is located.
Output	**Description**
Face Identifiers	Associate strings to elements belonging to some levels in a hierarchical classification (taxonomy).
Bounding Boxes	The box containing the Face identified.

10.7.4 JSON Metadata

https://schemas.mpai.community/PAF/V1.4/AIMs/FaceIdentityRecognition.json

10.7.5 Reference Software

10.7.5.1 Disclaimers

1. This PAF-FIR Reference Software Implementation is released with the BSD-3-Clause licence.
2. The purpose of this PAF-FIR Reference Software is to show a working Implementation of PAF-FIR, not to provide a ready-to-use product.
3. MPAI disclaims the suitability of the Software for any other purposes and does not guarantee that it is secure.
4. Use of this Reference Software may require acceptance of licences from the respective repositories. Users shall verify that they have the right to use any third-party software required by this Reference Software.

10.7.5.2 Guide to the PAF-FIR code

Use of this Reference Software for the PAF-FIR AI Module is for developers who are familiar with Python, Docker, RabbitMQ, and downloading models from HuggingFace

PAF-FIR performs face identity recognition with a pretrained FaceNet model; that is, it identifies the faces in a given number of frames per scene by comparison with a dataset of faces.

The PAF-FIR Reference Software is found at the MPAI gitlab site. It contains:

1. src: a folder with the Python code implementing the AIM
2. Dockerfile: a Docker file containing only the libraries required to build the Docker image and run the container
3. requirements.txt: dependencies installed in the Docker image
4. README.md: where to find and save weights of face recognition model FaceNet512.

Library: https://github.com/serengil/deepface

10.7.5.3 Acknowledgements

This version of the PAF-FIR Reference Software has been developed by the MPAI *AI Framework* Development Committee (AIF-DC).

10.7.6 Conformance Testing

Table 2 provides the Conformance Testing Method for PAF-FIR AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for PAF-FIR AIM

Receives	Text Object	Shall validate against Text Object Schema.
	Visual Object (Image)	Shall validate against Visual Object Schema. Image Data shall conform with Visual Qualifier.

	Face Time	Shall validate against Time Schema.
	Visual Scene Geometry	Shall validate against Visual Scene Geometry Schema.
Produces	Face Instance IDs	Shall validate against Instance ID Schema.
	Visual Object (Bounding Box)	Shall validate against Bounding Box Schema. Bounding Box Data shall conform with Visual Qualifier.

10.7.7 Performance Assessment

Performance Assessment of a PAF-FIR AIM Implementation shall be performed using a dataset of faces for each face of which the Identity of the face is provided with reference to a Taxonomy.

The Performance Assessment Report of an PAF-FIR AIM Implementation shall include:

1. The Identifier of the PAF-FIR AIM.
2. The identifier of the face dataset.
3. The identifier of the Taxonomy of face identifiers.

The Performance of the PAF-FIR AIM Implementation expressed by the Accuracy of the Identifiers provided by the output of the PAF-FIR AIM computed on all faces of the dataset referenced in 2 using the Taxonomy referenced.

10.8 Personal Status Display

10.8.1 Functions

Personal Status Display (PAF-PSD) V1.4 produces the Portable Avatar corresponding to an Avatar Model speaking a Text Object synthesised with a Speech Model and displaying a Personal Status:

Receives	Machine ID	ID to be used to identify the Avatar in Portable Avatar.
	Text Object	Text associated to Avatar in Portable Avatar.
	Personal Status	Personal Status associated to Avatar in Portable Avatar.
	Avatar Model	3D Model associated to Avatar in Portable Avatar.
	Speech Model	Speech Model Associated to Avatar in Portable Avatar.
Produces	Portable Avatar	Output Portable Avatar.
Enables	PAF-AVR	To render the Portable Avatar produced by PAF-PSD.

10.8.2 Reference Model

Figure 1 depicts the AIMs implementing the Personal Status Display (PAF-PSD) AIM.

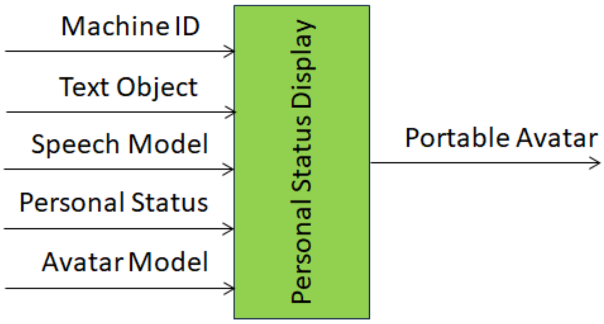

Figure 1 – Reference Model of Personal Status Display (PAF-PSD) AIM

10.8.3 Input/Output Data

Table 1 gives the Input/Output Data of Personal Status Display (PAF-PSD).

Table 1 – I/O Data of Personal Status Display

Input data	From	Description
Avatar ID	Upstream AIM	Portable Avatar's ID
Avatar Model	Upstream AIM or embedded in PSD	Part of Portable Avatar
Text Object	Keyboard or upstream AIM	Texts of Portable Avatar
Personal Status	Personal Status Extraction or Machine	To add PS to Speech, Face, and Gesture
Speech Model	Upstream AIM or embedded in PSD	Neural Network
Output data	**To**	**Description**
Portable Avatar	Downstream AIM or renderer	As Portable Avatar

10.8.4 SubAIMs

Figure 2 gives the Reference Model of the the Personal Status Display Composite AIM.

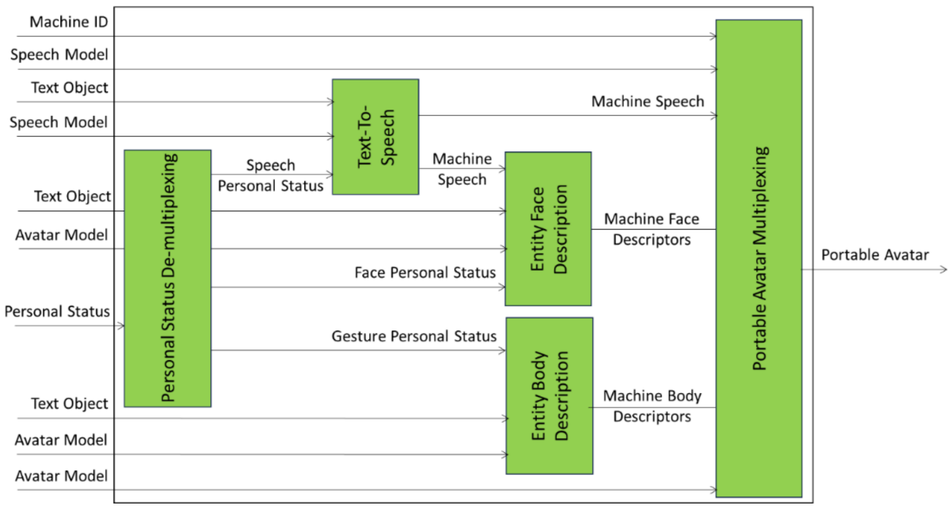

Figure 2 – Reference Model of Personal Status Display Composite AIM

The Personal Status Display Composite AIM operates as follows:

1. Avatar ID is the ID of the Portable Avatar.
2. Personal Status Demultiplexing makes available the component PS-Speech, PS-Face, and PS-Gesture Modalities.
3. Machine Text is synthesised as Speech using a Speech Model in a format specified by NN Format and the Personal Status provided by PS-Speech.
4. Machine Speech and PS-Face are used to produce the Machine Face Descriptors.
5. PS-Gesture and Text are used for Machine Body Descriptors using the Avatar Model.
6. Portable Avatar Multiplexing produces the Portable Avatar.

Table 2 gives the list of PSD AIMs with their input and output Data.

Table 2 –AIMs of Personal Status Display Composite AIM and JSON Metadata

AIW	AIMs	Name and Specification	JSON
PAF-PSD		Personal Status Display	X
	MMC-PDX	Personal Status Demultiplexing	X
	MMC-TTS	Text-to-Speech	X
	PAF-EFD	Entity Face Description	X
	PAF-EBD	Entity Body Description	X
	PAF-PMX	Portable Avatar Multiplexing	X

10.8.5 JSON Metadata

https://schemas.mpai.community/PAF/V1.4/AIMs/PersonalStatusDisplay.json

10.8.6 Profiles

The Profiles of Personal Status Display are specified.

10.8.7 Conformance Testing

The Conformance Testing Method for the PAF-PSD Basic AIM is provided here. The Conformance Testing Method for the individual Basic AIMs of the PAF-PSD Composite AIM is provided by the individual Basic AIMs.

Table 2 provides the Conformance Testing Method for PAF-PSD AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for PAF-PSD AIM

Receives	Machine ID	Shall be string or validate against Instance ID Schema
	Text Object	Shall validate against Text Object Schema. Text Data shall conform with Speech Qualifier.

	Personal Status	Shall validate against Personal Status Schema.
	Avatar Model	Shall validate against 3D Model Schema. Avatar Model Data shall conform with 3D Model Qualifier.
	Speech Model	Shall validate against Machine Learning Model Schema. Speech Model Data shall conform with Machine Learning Model Qualifier.
Produces	Portable Avatar	Shall validate against Portable Avatar Schema. Portable Avatar Data shall conform with respective Qualifiers.

10.9 Personal Status Extraction

10.9.1 Functions

Personal Status Extraction (MMC-PSE):

Receives	*Text Object* or *Text Descriptors*	
	Text Selector	indicating whether Text or Text Descriptors should be used.
	Speech Object or *Speech Descriptors*	
	Speech Selector	indicating whether Speech or Speech Descriptors should be used.
	Face or *Face Descriptors*	
	Face Selector	indicating whether Face or Face Descriptors should be used.
	Body or *Gesture Descriptors*	
	Body Selector	indicating whether Body or Gesture Descriptors should be used.
Computes and then Interprets	depending on reception of	the Descriptors of a Modality (Text, Speech, or Face).
	Text Descriptors	alternatively, Interprets the received Descriptors and produces Personal Status of the Text Object (PS-Text).
	Speech Descriptors;	alternatively, Interprets the received Descriptors and produces Personal Status of the Speech Object (PS-Speech).
	Face Descriptors	alternatively, Interprets the received Descriptors and produces Personal Status of the Face (PS-Face).
	Gesture Descriptors	alternatively, Interprets the received Gesture Descriptors of the Body.
Multiplexes	The results of the interpretations.	
Produces	Entity's Personal Status	

10.9.2 Reference Model

Figure 1 depicts the Reference Model of the Personal Status Extraction (MMC-PSE) AIM.

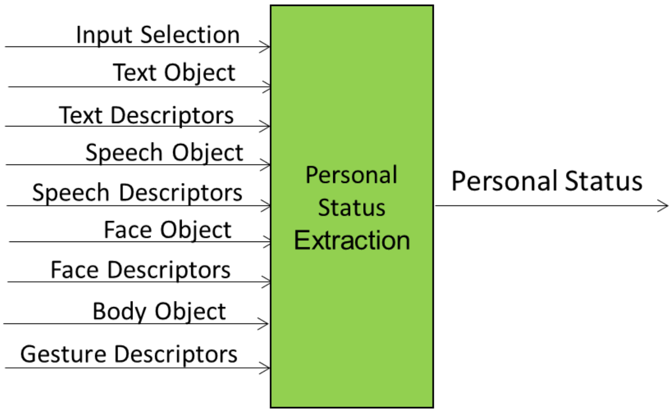

Figure 1 – The Personal Status Extraction Composite (MMC-PSE) AIM Reference Model

10.9.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Personal Status Extraction (MMC-PSE) AIM.

Table 1 – I/O Data of the Personal Status Extraction (MMC-PSE) AIM

Input data	From	Description
Input Selector	An external signal	Media or Descriptors Selector
Text Object	Keyboard or AIM	Text or Recognised Text.
Text Descriptors	An upstream AIM	Functionally equivalent to Text Description.
Speech Object	Microphone/upstream AIM	Speech of Entity.
Speech Descriptors	An upstream AIM	Functionally equivalent to Speech Description.
Face Visual Object	Visual Scene Description	The face of the Entity.
Face Descriptors	An upstream AIM	Functionally equivalent to Face Description.
Body Visual Object	Visual Scene Description	The body of the Entity.
Gesture Descriptors	An upstream AIM	Functionally equivalent to Body Description.
Output data	**To**	**Description**
Personal Status	A downstream AIM	For further processing

10.9.4 SubAIMs

A Personal Status Extraction AIM instance can be implemented as a Composite AIM with different degrees of composition. The most extended composition if depicted by Figure 2

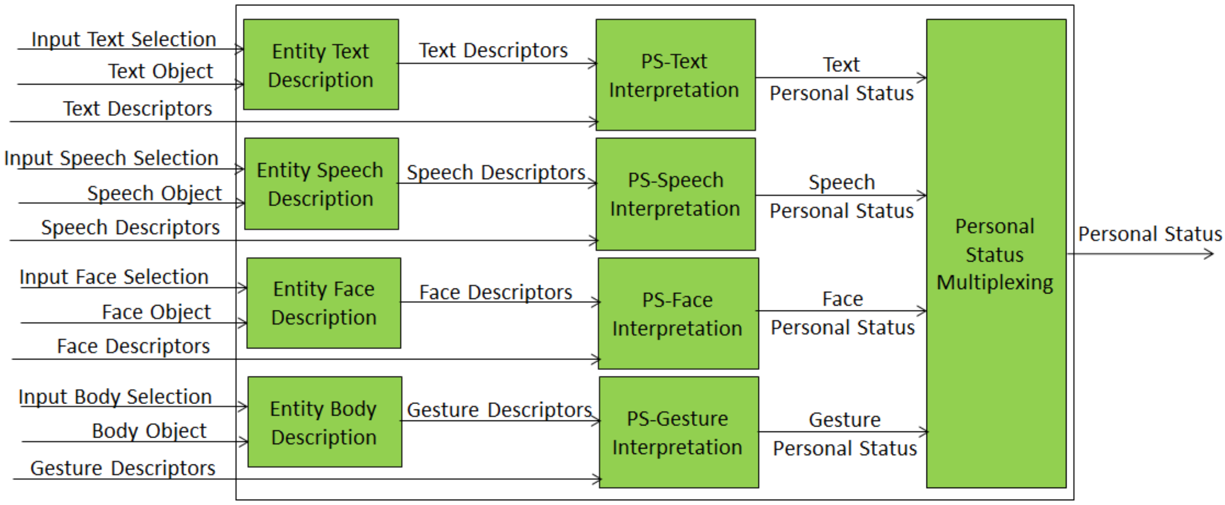

Figure 2 - The version of Personal Status Extraction AIM with the highest level of composition.

Table 2 gives the AIMs and their JSON Metadata of MMC-PSE.

Table 2 - AIMs and JSON Metadata

AIMs	AIMs	AIM Names	JSON
MMC-PSE		Personal Status Extraction	X
	MMC-ETD	Entity Text Description	X
	MMC-ESD	Entity Speech Description	X
	PAF-EFD	Entity Face Description	X
	PAF-EBD	Entity Body Description	X
	MMC-PTI	PS-Text Interpretation	X
	MMC-PSI	PS-Speech Interpretation	X
	PAF-PFI	PS-Face Interpretation	X
	PAF-PGI	PS-Gesture Interpretation	X
	MMC-PMX	Personal Status Multiplexing	X

10.9.5 JSON Metadata

https://schemas.mpai.community/MMC/V2.3/AIMs/PersonalStatusExtraction.json

10.9.6 Profiles

The Profiles of Personal Status Extraction are specified.

10.9.7 Conformance Testing

Table 3 provides the Conformance Testing Method for MMC-PSE AIM as a Basic AIM. Conformance Testing of the individual AIMs of the MMC-PSE Composite AIM are given by the individual AIM Specification.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data that refers to a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 3 – Conformance Testing Method for MMC-PSE AIM

Input	Text Object or	Shall validate against Text Object schema. Text Data shall conform with Text Qualifier.
	Text Descriptors	Shall validate against Text Descriptors schema.
	Text Selector	Shall validate against Text Selector schema.
	Speech Object or	Shall validate against Speech Object schema. Speech Data shall conform with Speech Qualifier.
	Speech Descriptors	Shall validate against Speech Descriptors schema.
	Speech Selector	Shall validate against Speech Selector schema.
	Face Visual Object or	Shall validate against Visual Object schema. Visual Data shall conform with Visual Qualifier.
	Face Descriptors	Shall validate against Face Descriptors schema.
	Face Selector	Shall validate against Face Selector schema.
	Body Visual Object	Shall validate against Visual Object schema. Visual Data shall conform with Visual Qualifier.
	Gesture Descriptors	Shall validate against Gesture Descriptors schema.
	Body Selector	Shall validate against Body Selector schema.
Output	Entity Personal Status	Shall validate against Personal Status schema.

10.10 Speaker Identity Recognition

10.10.1 Functions

Speaker Identity Recognition (MMC-SIR):

Receives	*Auxiliary Text*	Text related to the Speech.
	Speech Object	Speech of which the Speaker is requested.
	Speech Time	Time during whose duration Speaker ID is requested.
	Speech Overlap	Data signalling which parts of Speech Data have overlapping speech.
	Speech Scene Geometry	Disposition of Speech Data of the scene where the Speech whose speaker is to be identified is located.
Produces	*Speaker Identifier*	ID of speaker.

10.10.2Reference Model

The Reference Architecture of Speaker Identity Recognition (MMC-SIR) is depicted in Figure 1.

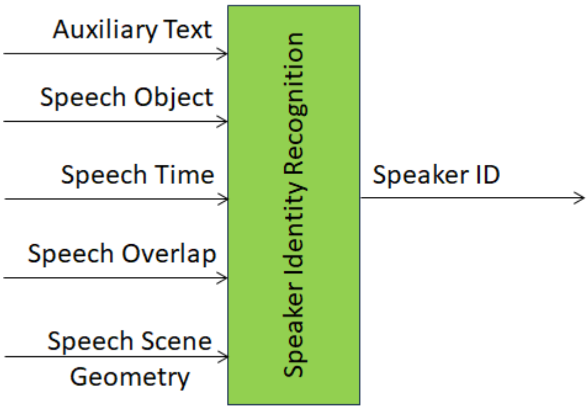

Figure 1 – The Speaker Identity Recognition (MMC-SIR) AIM

10.10.33 Input/Output Data

Table 1 specifies the Input and Output Data of the Speaker Identity Recognition (MMC-SIR) AIM.

Table 1 – I/O Data of the Speaker Identity Recognition (MMC-SIR) AIM

Input	Description
Auxiliary Text	Text with content related to Speaker ID.
Speech Object	Speech Object emitted by the Speaker.
Speech Time	The start and end time of the Speech.
Speech Overlap	Information about overlapping Speech.
Speech Scene Geometry	Information about Speech Object location.
Output	**Description**
Speaker Identifier	The Visual Descriptors of the Visual Scene.

10.10.4JSON Metadata

https://schemas.mpai.community/MMC/V2.3/AIMs/SpeakerIdentityRecognition.json

10.10.5Reference Software

10.10.5.1 Disclaimers

1. This MMC-SIR Reference Software Implementation is released with the BSD-3-Clause licence.
2. The purpose of this MMC-SIR Reference Software is to show a working Implementation of MMC-SIR, not to provide a ready-to-use product.
3. MPAI disclaims the suitability of the Software for any other purposes and does not guarantee that it is secure.
4. Use of this Reference Software may require acceptance of licences from the respective repositories. Users shall verify that they have the right to use any third-party software required by this Reference Software.

10.10.5.2 Guide to the MMC-SIR code

MMC-SIR performs speaker verification with a pretrained ECAPA-TDNN model; that is, it identifies the speaker of each speech segment by comparison with a dataset consisting of short clips of human speech.

The MMC-SIR Reference Software is found at the MPAI gitlab site. It contains:

1. src: a folder with the Python code implementing the AIM
2. Dockerfile: a Docker file containing only the libraries required to build the Docker image and run the container
3. requirements.txt: dependencies installed in the Docker image
4. README.md: commands for cloning https://huggingface.co/speechbrain/spkrec-ecapa-voxceleb

Library: https://github.com/speechbrain/speechbrain

10.10.5.3 Acknowledgements

This version of the MMC-SIR Reference Software has been developed by the MPAI *AI Framework* Development Committee (AIF-DC).

10.11 Visual Object Identification

10.11.1Functions

Visual Object Identification (OSD-VOI) V1.3 identifies a Visual Object included in a Visual Scene Geometry by providing the Point of View:

Receives	*Visual Scene Geometry*	The arrangement of the objects in the Scene, a subset of Visual Scene Descriptors.
	Visual Objects	The Objects in the Scene.
	Body Descriptors	Descriptors of the Body indicating the object.
Produces	*Visual Instance ID*	Identifying a Visual Object in the Scene that belongs to some level in a taxonomy.

10.11.2Reference Model

Figure 1 specifies the Reference Model of Visual Object Identification (OSD-VOI) AIM.

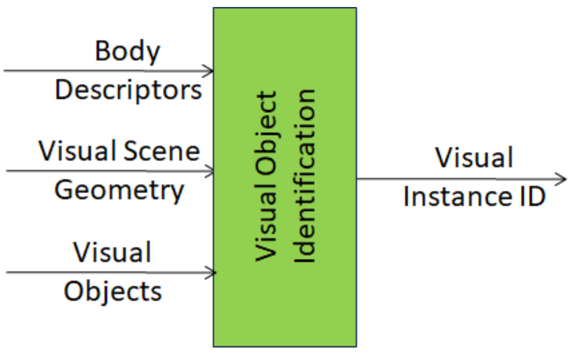

Figure - The Visual Object Identification (OSD-VOI) AIM Reference Model

10.11.3Input/Output Data

Table 1 specifies the Input and Output Data of the Visual Object Identification (OSD-VOI) AIM.

Table 1 – I/O Data of the Visual Object Identification (OSD-VOI) 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.
Visual Object	The Visual Objects in the Visual Scene that are not Entities.
Output	**Description**
Visual Instance Identifier	The Identifier of the specific Visual Object belonging to a level in the taxonomy.

10.11.4SubAIMs

Visual Object Identification (OSD-VOI) is a Composite AIM specified by Figure 2.

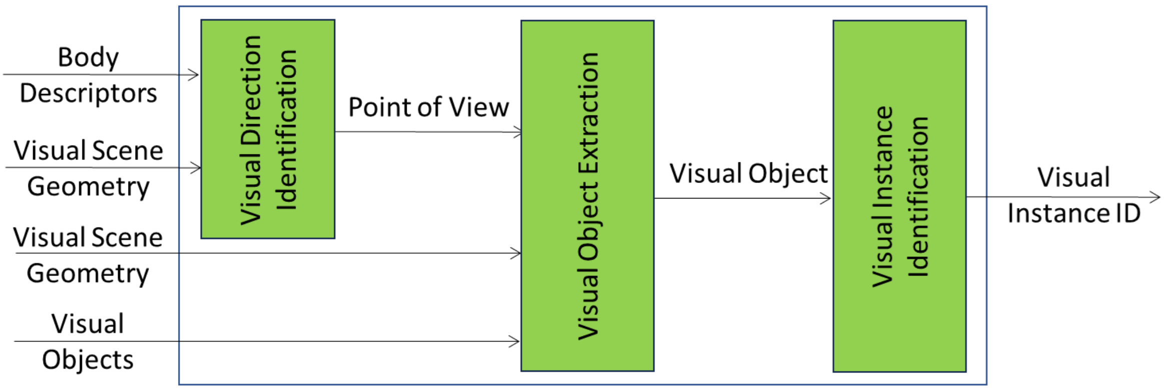

Figure 2 - The Visual Object Identification (OSD-VOI) Composite AIM

Note that the Visual Direction Identification AIM can parse either an AV Scene Geometry or its Visual Scene Geometry subset.

The AIMs composing the Visual Object Identification (OSD-VOI) Composite AIM are:

AIM	AIMs	Names	JSON
OSD-VOI		Visual Object Identification	Link
	OSD-VDI	Visual Direction Identification	Link
	OSD-VOE	Visual Object Extraction	Link
	OSD-VII	Visual Instance Identification	Link

10.11.5JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/VisualObjectIdentification.json

10.11.6Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-VOI AIM. Conformance Testing of the individual AIMs of the OSD-VOI Composite AIM are given by the individual AIM Specification.

Note that a schema may contain references to other schemas. In this case, validation of data for the primary schema implies that any data that refers to a secondary schema shall also validate.

Table 2 – Conformance Testing Method for OSD-VOI AIM

Receives	Visual Scene Geometry	Shall validate against Visual Scene Geometry schema.
	Visual Objects	Shall validate against Visual Objects schema. Visual Data shall conform with Qualifier.
	Body Descriptors	Shall validate against Body Descriptors XML schema.
Produces	Visual Instance ID	Shall validate against Instance ID schema.

10.12 Audio Scene Description

10.12.1 Functions

Audio Scene Description (OSD-ASD) V1.3 produces the Descriptors of a Scene composed by Audio Objects and Scenes:

Receives	Space-Time	of the input Objects having the same time base.
	Audio Objects	individual Audio Objects.
	Scene Descriptors	Scene to Objects belong to.
Integrates	Space-Time and 3D Model Object	with Scene Descriptors.
Produces	Audio Scene Descriptors	Output#1 of AIM
	Alert	Output#2 of AIM signalling potential anomalies in Object.

10.12.2Reference Model

The Reference Architecture is depicted in Figure 1.

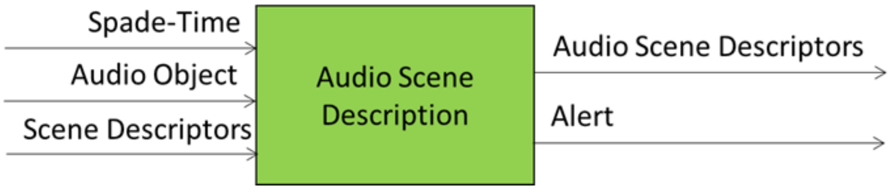

Figure 1 – The Audio Scene Description (OSD-ASD) AIM

10.12.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Audio Scene Description (OSD-ASD) AIM. Links are to the Data Type specifications.

Table 1 – I/O Data of the Audio Scene Description (OSD-ASD) AIM

Input	Description
Space-Time	Space-Time of input Objects.
Audio Objects	Input Objects.
Scene Descriptors	Input Scene Descriptors.
Output	**Description**
Audio Scene Descriptors	The output Audio Scene Descriptors.
Alert	Data signalling potential anomalies in Object.

10.12.4 JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/SceneDescription.json

10.12.5 Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-3SD AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-3SD AIM

Receives	Space-Time	Shall validate against Space-Time schema.
	Audio Objects	Shall validate against Audio Object schema. Media-specific Data shall conform with their Qualifiers.
	Scene Descriptors	Shall validate against Scene Descriptors schema.
Produces	Audio Scene Descriptors	Shall validate against Audio Scene Descriptors schema.
	Alert	Shall validate against Alert schema.

10.13 Basic Environment Description

10.13.1 Functions

The Basic Environment Descriptors (CAV-BED) V1.0 AIM produces the Basic Scene Descriptors from Audio, LiDAR, RADAR, Ultrasound, and Visual Scene Descriptors, :

Receives	Audio Scene Descriptors	From Audio Scene Description.
	LiDAR Scene Descriptors	LiDAR Scene Description.

	RADAR Scene Descriptors	RADAR Scene Description.
	Offline Map Scene Descriptors.	Offline Map Scene Description.
	Ultrasound Scene Descriptors	Ultrasound Scene Description
	Visual Scene Descriptors	Visual Scene Description.
	Weather Data	From Motion Actuation Subsystem.
	Full Environment Descriptors	From Autonomous Motion Subsystem.
Produces	Basic Environment Descriptors	To Autonomous Motion Subsystem.

10.13.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the Basic Environment Description AIM.

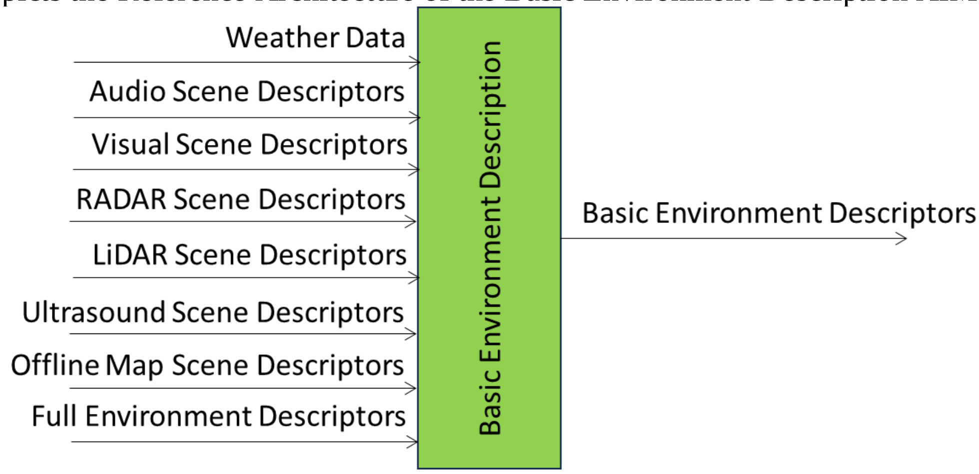

Figure 1 – The Basic Environment Description AIM

10.13.3 I/O Data

Table 1 specifies the Input and Output Data of the Basic Environment Description AIM.

Table 1 – I/O Data of the Basic Environment Description AIM

Input Data	Description
Audio Scene Descriptors	Descriptors from Audio Scene Description AIM.
LiDAR Scene Descriptors	Descriptors from LiDAR Scene Description AIM.
RADAR Scene Descriptors	Descriptors from RADAR Scene Description AIM.
Offline Map Scene Descriptors	Descriptors from Offline Map Scene Description AIM.
Ultrasound Scene Descriptors	Descriptors from Ultrasound Scene Description AIM.
Visual Scene Descriptors	Descriptors from Visual Scene Description AIM.
Weather Data	Weather Data from Motion Actuation Subsystem.
Full Environment Descriptors	From the Autonomous Motion Subsystem.
Output Data	**Description**
Basic Environment Descriptors	Environment Sensing Subsystem's Basic Environment Descriptors.

.

10.13.4 JSON Metadata

https://schemas.mpai.community/CAV2/V2.0/AIMs/BasicEnvironmentDescription.json

10.14 LiDAR Scene Description

10.14.1Functions

LiDAR Scene Description (OSD-LSD) V1.3 produces the Descriptors of a Scene composed by LiDAR Objects and Scenes:

Receives	Space-Time	of the input Objects having the same time base.
	LiDAR Objects	Individual LiDAR Objects.
	Scene Descriptors	Scene the Objects belong to.
Integrates	Space-Time and LiDAR Object	with Scene Descriptors.
Produces	LiDAR Scene Descriptors	Output#1 of AIM
	Alert	Output#2 of AIM signalling potential anomalies in Object.

10.14.2Reference Model

The Reference Architecture is depicted in Figure 1.

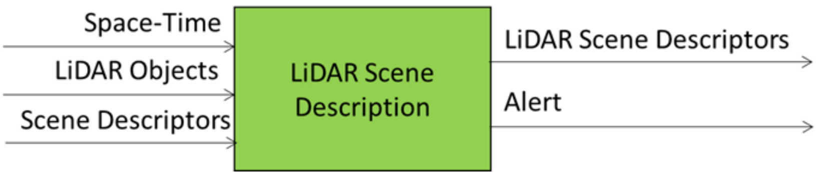

Figure 1 – The LiDAR Scene Description (OSD-LSD) AIM

10.14.3Input/Output Data

Table 1 specifies the Input and Output Data of the LiDAR Scene Description (OSD-LSD) AIM. .

Table 1 – I/O Data of the LiDAR Scene Description (OSD-LSD) AIM

Input	Description
Space-Time	Space-Time of input Objects.
LiDAR Objects	Input LiDAR Objects.
Scene Descriptors	Input Scene Descriptors.
Output	**Description**
LiDAR Scene Descriptors	The output LiDAR Descriptors.
Alert	Data signaling potential anomalies in Object.

10.14.4JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/LiDARSceneDescription.json

10.14.5 Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-LSD AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-3SD AIM

Receives	Space-Time	Shall validate against Space-Time schema.
	LiDAR Objects	Shall validate against LiDAR Object schema. Media-specific Data shall conform with their Qualifiers.
	Scene Descriptors	Shall validate against Scene Descriptors schema.
Produces	LiDAR Scene Descriptors	Shall validate against LiDAR Scene Descriptors schema.
	Alert	Shall validate against Alert schema.

10.15 Offline Map Scene Description

10.15.1 Functions

Offline Map Scene Description (OSD-OSD) V1.3 produces the Descriptors of a Scene composed by Offline Map Objects and Scenes:

Receives	Space-Time	of the input Objects having the same time base.
	Offline Map Objects	Individual Offline Map Objects.
	Scene Descriptors	Scene the Objects belong to.
Integrates	Space-Time and Offline Map Object	with Scene Descriptors.
Produces	Offline Map Scene Descriptors	Output#1 of AIM
	Alert	Output#2 of AIM signalling potential anomalies in Object.

10.15.2 Reference Model

The Reference Architecture is depicted in Figure 1.

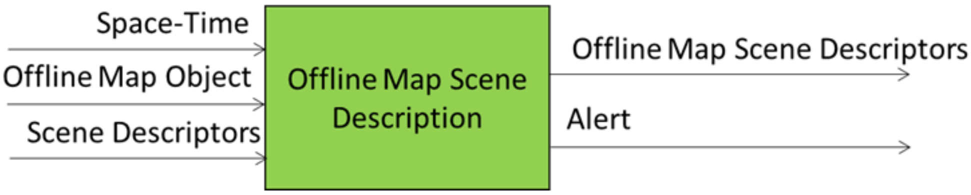

Figure 1 – The Offline Map Scene Description (OSD-OSD) AIM

10.15.3Input/Output Data

Table 1 specifies the Input and Output Data of the Offline Map Scene Description (OSD-OSD) AIM.

Table 1 – I/O Data of the Offline Map Scene Description (OSD-OSD) AIM

Input	Description
Space-Time	Space-Time of input Objects.
Offline Map Objects	Input Offline Map Objects.
Scene Descriptors	Input Scene Descriptors.
Output	**Description**
Offline Map Scene Descriptors	The output Offline Map Scene Descriptors.
Alert	Data signalling potential anomalies in Object.

10.15.45 JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/OfflineMapSceneDescription.json

10.15.5Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-OSD AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-OSD AIM

Receives	Space-Time	Shall validate against Space-Time schema.
	Offline Map Objects	Shall validate against Offline Map Object schema. Media-specific Data shall conform with their Qualifiers.
	Scene Descriptors	Shall validate against Scene Descriptors schema.
Produces	Offline Map Scene Descriptors	Shall validate against Offline Map Scene Descriptors schema.
	Alert	Shall validate against Alert schema.

10.16 RADAR Scene Description

10.16.1Functions

RADAR Scene Description (OSD-RSD) V1.3 produces the Descriptors of a Scene composed by RADAR Objects and Scenes:

Receives	Space-Time	of the input Objects having the same time base.
	RADAR Objects	Individual RADAR Objects.
	Scene Descriptors	Scene the Objects belong to.
Integrates	Space-Time and RADAR Object	with Scene Descriptors.
Produces	RADAR Scene Descriptors	Output#1 of AIM
	Alert	Output#2 of AIM signalling potential anomalies in Object.

10.16.2 Reference Model

The Reference Architecture is depicted in Figure 1.

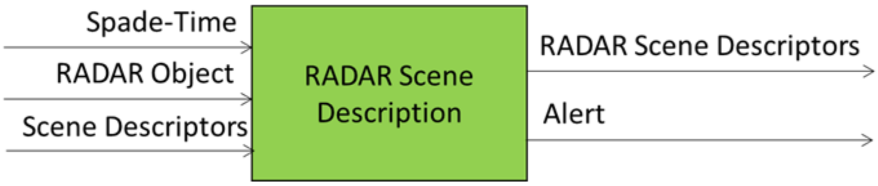

Figure 1 – The RADAR Scene Description (OSD-RSD) AIM

10.16.3 Input/Output Data

Table 1 specifies the Input and Output Data of the RADAR Scene Description (OSD-RSD) AIM. .

Table 1 – I/O Data of the RADAR Scene Description (OSD-RSD) AIM

Input	Description
Space-Time	Space-Time of input Objects.
RADAR Objects	Input RADAR Objects.
Scene Descriptors	Input Scene Descriptors.
Output	**Description**
RADAR Scene Descriptors	The output RADAR Scene Descriptors.
Alert	Data signalling potential anomalies in Object.

10.16.4 JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/RADARSceneDescription.json

10.16.5 Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-RSD AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-RSD AIM

Receives	Space-Time	Shall validate against Space-Time schema.
	RADAR Objects	Shall validate against RADAR Object schema. Media-specific Data shall conform with their Qualifiers.
	Scene Descriptors	Shall validate against Scene Descriptors schema.
Produces	RADAR Scene Descriptors	Shall validate against RADAR Scene Descriptors schema.
	Alert	Shall validate against Alert schema.

10.17 Spatial Attitude Generation

10.17.1 Functions

The Spatial Attitude Generation (CAV-SAG) AIM:

Receives	*GNSS Object*	*From GNSS receiver.*
	Spatial Attitude	*From initial Spatial Attitude computed by Motion Actuation Subsystem.*
Integrates	*The two data streams*	*GNSS and Initial Spatial Attitude.*
Produces	*Spatial Attitude*	*The CAV's reference Spatial Attitude.*

10.17.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the Spatial Attitude Generation (CAV-SAG) AIM.

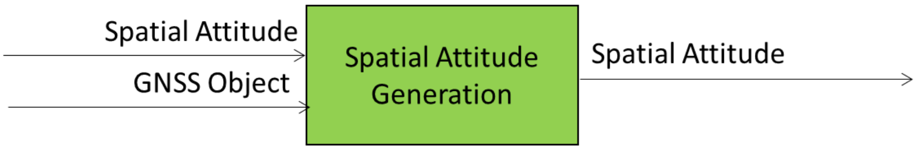

Figure 1 – The Spatial Attitude Generation (CAV-SAG) AIM

10.17.3I/O Data

Table 1 specifies the Input and Output Data of the Spatial Attitude Generation (CAV-SAG) AIM.

Table 1 – I/O Data of the Spatial Attitude Generation (CAV-SAG) AIM

Input Data	Description
Spatial Attitude	Initial estimate of CAV's Spatial Attitude From MAS.
GNSS Object	GNSS Data and Qualifier

Output Data	Description
Spatial Attitude	Final Spatial Attitude

10.17.4 JSON Metadata

https://schemas.mpai.community/CAV1/V1.1/AIMs/SpatialAttitudeGeneration.json

10.18 Ultrasound Scene Description

10.18.1Functions

Ultrasound Scene Description (OSD-USD) V1.3 produces the Descriptors of a Scene composed by Ultrasound Objects and Scenes:

Receives	Space-Time	of the input Objects having the same time base.
	Ultrasound Objects	Individual Ultrasound Objects.
	Scene Descriptors	Scene the Objects belong to.
Integrates	Space-Time and Ultrasound Object	with Scene Descriptors.
Produces	Ultrasound Scene Descriptors	Output#1 of AIM
	Alert	Output#2 of AIM signalling potential anomalies in Object.

10.18.2Reference Model

The Reference Architecture is depicted in Figure 1.

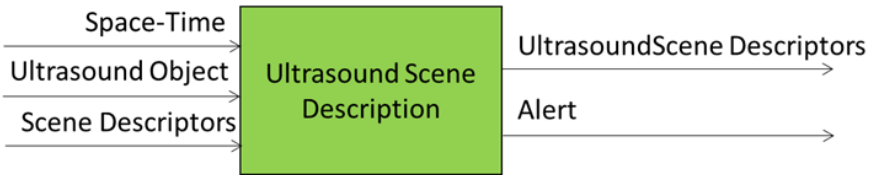

Figure 1 – The Ultrasound Scene Description (OSD-USD) AIM

10.18.3Input/Output Data

Table 1 specifies the Input and Output Data of the Ultrasound Scene Description (OSD-USD) AIM.

Table 1 – I/O Data of the Ultrasound Scene Description (OSD-USD) AIM

Input	Description
Space-Time	Space-Time of input Objects.
Ultrasound Objects	Input Ultrasound Objects.
Scene Descriptors	Input Scene Descriptors.
Output	**Description**

Ultrasound Scene Descriptors	The output Ultrasound Scene Descriptors.
Alert	Data signalling potential anomalies in Object.

10.18.4 JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/UltrasoundSceneDescription.json

10.18.5 Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-USD AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-USD AIM

Receives	Space-Time	Shall validate against Space-Time schema.
	Ultrasound Objects	Shall validate against Ultrasound Object schema. Media-specific Data shall conform with their Qualifiers.
	Scene Descriptors	Shall validate against Scene Descriptors schema.
Produces	Ultrasound Scene Descriptors	Shall validate against Ultrasound Scene Descriptors schema.
	Alert	Shall validate against Alert schema.

10.19 Visual Scene Description

10.19.1 Functions

Visual Scene Description (OSD-VSD) V1.3 produces the Descriptors of a Scene composed by Visual Objects and Scenes:

Receives	Space-Time	of the input Objects having the same time base.
	Visual Objects	Individual Visual Objects.
	Scene Descriptors	Scene the Objects belong to.
Integrates	Space-Time and Visual Object	with Scene Descriptors.
Produces	Visual Scene Descriptors	Output#1 of AIM
	Alert	Output#2 of AIM signalling potential anomalies in Object.

10.19.2 Reference Model

The Reference Architecture is depicted in Figure 1.

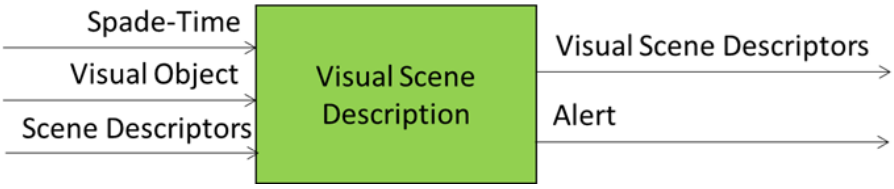

Figure 1 – The Visual Scene Description (OSD-VSD) AIM

10.19.3 Input/Output Data

Table 1 specifies the Input and Output Data of the Visual Scene Description (OSD-VSD) AIM.

Table 1 – I/O Data of the Visual Scene Description (OSD-VSD) AIM

Input	Description
Space-Time	Space-Time of input Objects.
Visual Objects	Input Visual Objects.
Scene Descriptors	Input Scene Descriptors.
Output	**Description**
Visual Scene Descriptors	The output Visual Scene Descriptors.
Alert	Data signalling potential anomalies in Object.

10.19.4 JSON Metadata

https://schemas.mpai.community/OSD/V1.3/AIMs/VisualSceneDescription.json

10.19.5 Conformance Testing

Table 2 provides the Conformance Testing Method for OSD-VSD AIM.

If a schema contains references to other schemas, conformance of data for the primary schema implies that any data referencing a secondary schema shall also validate against the relevant schema, if present and conform with the Qualifier, if present.

Table 2 – Conformance Testing Method for OSD-VSD AIM

Receives	Space-Time	Shall validate against Space-Time schema.
	Visual Objects	Shall validate against Visual Object schema. Media-specific Data shall conform with their Qualifiers.
	Scene Descriptors	Shall validate against Scene Descriptors schema.
Produces	Visual Scene Descriptors	Shall validate against Visual Scene Descriptors schema.
	Alert	Shall validate against Alert schema.

10.20 AMS Memory

10.20.1 Functions

The AMS Memory (CAV-AMM) AIM:

Receives	*Full Environment Descriptors*	From Full Environment Description
	Route	From Route Selection Planning
	Path	From Motion Planning and Decision/Path Selection Planning
	Trajectory	From Motion Planning and Decision/Path Selection Planning
	Alert	From Traffic Obstacle Avoidance
	Road State	From AMS-MAS Message Issuance
	CAV State	From AMS-MAS Message Issuance
	AMS-MAS Message	From AMS-MAS Message Issuance, including sent and received messages
Produces	*AMS Data*	To all AIMs in AMS.

10.20.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the AMS Memory (CAV-AMM) AIM.

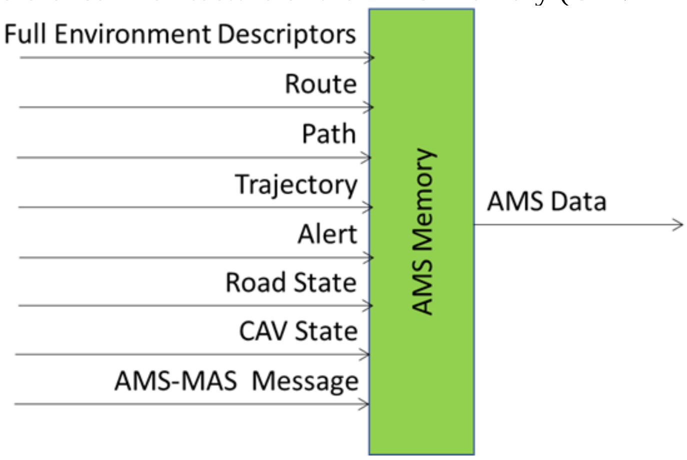

Figure 1 – The AMS Memory (CAV-AMM) AIM

10.20.3 I/O Data

Table 1 specifies the Input and Output Data of the AMS Memory (CAV-AMM) AIM.

Table 1 – I/O Data of the AMS Memory (CAV-AMM) AIM

Input	Description
Full Environment Descriptors	Full Environment Description.
Route	From CAV-FED.
Path	Form CAV-PSP.
Trajectory	From CAV-MSP.
Alert	From CAV-TPD.
Road State	From CAV-AMI.

CAV State	From CAV-AMI.
AMS-MAS Message	To/from Motion Actuation Subsystem.
Output	**Description**
AMS Data	AMS Recording Data for use by external Device.

10.20.4JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/AMSMemory.json

10.21 Full Environment Description

10.21.1 Functions

The Full Environment Description (CAV-FED) AIM:

Receives	*Basic Environment Descriptors*	From Environment Sensing Subsystem.
	Full Environment Descriptors	From CAVs in range.
	Road State	From AMS-MAS Message Issuance received from MAS.
	CAV State	From AMS-MAS Message Response.
	AMS Data	CAV Data from AMS Memory AIM.
Creates	*Internal environment representation*	By fusing information received from ESS, Remote AMSs, and other CAV-aware entities.
Updates	*CAV State*	Managed by AMS.
Produces	*Full Environment Descriptors*	To Route, Path, Motion Selection Planning, Obstacle Traffic Avoidance, and AMS Decision Recording.

10.21.2Reference Architecture

Figure 1 depicts the Reference Architecture of the Full Environment Description AIM.

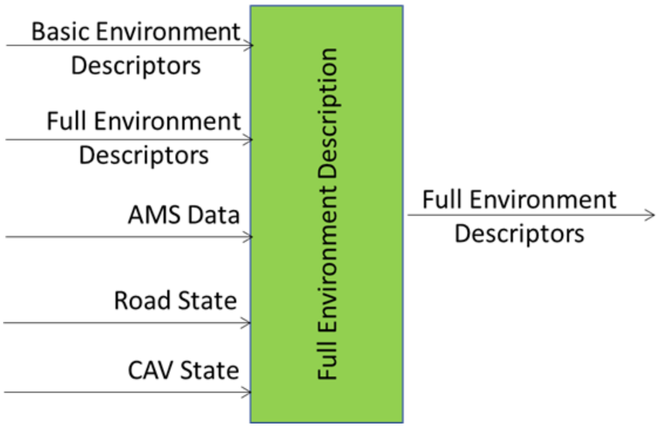

Figure 1 – The Full Environment Description AIM

10.21.3 I/O Data

Table 1 specifies the Input and Output Data of the Full Environment Description AIM.

Table 1 – I/O Data of the Full Environment Description AIM

Input	Description
Basic Environment Descriptors	From ESS.
Full Environment Descriptors	From Remote CAVs.
AMS Data	CAV Data from AMS Memory AIM.
CAV State	From AMS-MAS Message.
Road State	From MAS Command Issuer, forwarding Road State from MAS.
Output	**Description**
Full Environment Descriptors	Ego CAV's Full Environment Descriptors to AMS internal AIMs and the ESS.

10.21.4 JSON Metadata

https://schemas.mpai.community/CAV1/V1.1/AIMs/FullEnvironmentDescription.json

10.22 Motion Selection Planning

10.22.1 Functions

The Motion Selection Planning (CAV-MSP) AIM:

Receives	*Full Environment Descriptors*	From Full Environment Description.
	AMS Data	CAV Data from AMS Memory.
	Path	From Path Selection Planning.
Produces	*Trajectory*	To Traffic Obstacle Avoidance and AMS Memory.

10.22.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the Motion Selection Planning (CAV-MSP) AIM.

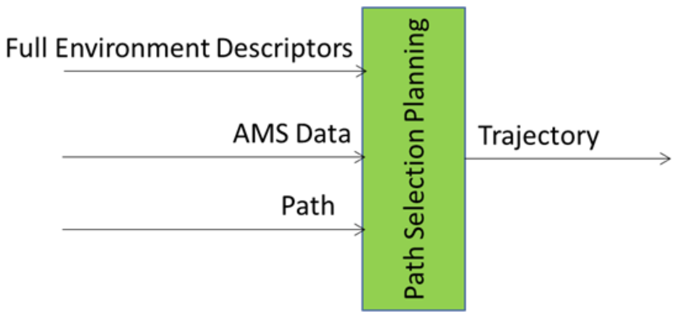

Figure 1 – The Motion Selection Planning (CAV-MSP) AIM

10.22.3 I/O Data

Table 1 specifies the Input and Output Data of the Motion Selection Planning (CAV-MSP) AIM.

Table 1 – I/O Data of the Motion Selection Planning (CAV-MSP) AIM

Input	Description
Full Environment Descriptors	From Full Environment Description AIM.
AMS Data	
Path	From Path Selection Planning AIM.
Output	**Description**
Trajectory	Passed to the Traffic Obstacle Avoidance AIM.

10.22.4 JSON Metadata

https://schemas.mpai.community/CAV1/V1.1/AIMs/MotionSelectionPlanning.json

10.23 Path Selection Planning

10.23.1 Functions

The Path Selection Planning (CAV-PSP) AIM:

Receives	*Full Environment Descriptors*	From Full Environment Description.
	AMS Data	From AMS Memory
	Route	From Route Selection Planning.
Produces	*Path*	To Motion Selection Planning.

10.23.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the Path Selection Planning AIM.

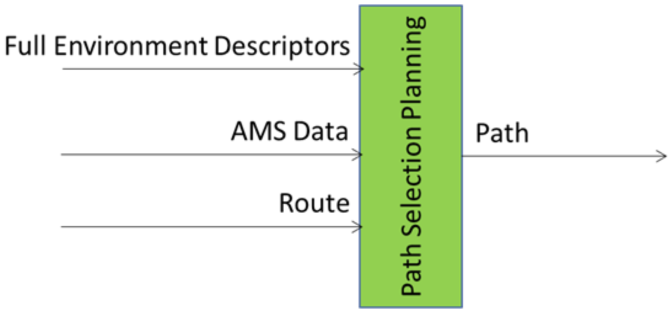

Figure 1 – The Path Selection Planning AIM

10.23.3I/O Data

Table 1 specifies the Input and Output Data of the Path Selection Planning AIM.

Table 1 – I/O Data of the Path Selection Planning AIM

Input	Description
Full Environment Descriptors	Provided by the CAV-FED.
AMS Data	CAV Data from AMS Memory.
Route	To AMS Decision Recording.
Output	**Description**
Path	Motion passed to the Path Selection Planning AIM.

10.23.4 JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/PathSelectionPlanning.json

10.24 Route Selection Planning

10.24.1 Functions

The Route Selection Planning (CAV-RSP) AIM:

Receives	*Full Environment Descriptors*	From Full Environment Description.
	AMS Data	From AMS Memory.
	AMS-HCI Message	From Human-Machine Interaction AIW.
Produces	*AMS-HCI Message*	To Human-Machine Interaction AIW.
	Route	To Path Selection Planning for implementation and AMS Decision Recording.

10.24.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the Full Environment Description AIM.

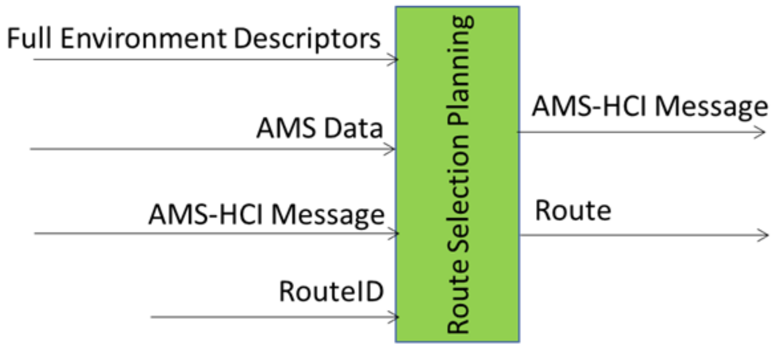

Figure 1 – The Route Selection Planning AIM

10.24.3 I/O Data

Table 1 specifies the Input and Output Data of the Route Selection Planning AIM.

Table 1 – I/O Data of the Route Selection Planning AIM

Input	Description
Full Environment Descriptors	Provided by the CAV-FED.
AMS Data	CAV Data from AMS Memory.
AMS-HCI Message	Request message received from HCI.
Route ID	Selected Route represented by its ID.
Output	**Description**
AMS-HCI Message	Request sent to HCI.
Route	Route passed to the Path Selection Planning AIM

10.24.4 JSON Metadata

https://schemas.mpai.community/CAV1/V1.1/AIMs/RouteSelectionPlanning.json

10.25 Spatial Attitude Generation

10.25.1 Functions

The Spatial Attitude Generation (CAV-SAG) AIM:

Receives	*GNSS Object*	*From GNSS receiver.*
	Spatial Attitude	*From initial Spatial Attitude computed by Motion Actuation Subsystem.*
Integrates	*The two data streams*	*GNSS and Initial Spatial Attitude.*
Produces	*Spatial Attitude*	*The CAV's reference Spatial Attitude.*

10.25.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the Spatial Attitude Generation (CAV-SAG) AIM.

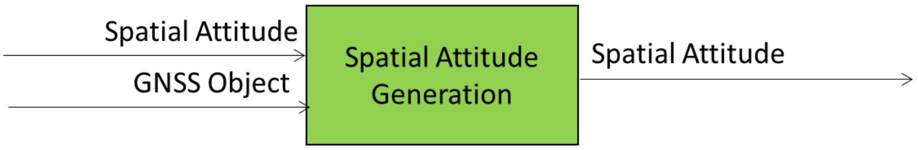

Figure 1 – The Spatial Attitude Generation (CAV-SAG) AIM

10.25.3I/O Data

Table 1 specifies the Input and Output Data of the Spatial Attitude Generation (CAV-SAG) AIM.

Table 1 – I/O Data of the Spatial Attitude Generation (CAV-SAG) AIM

Input Data	Description
Spatial Attitude	Initial estimate of CAV's Spatial Attitude From MAS.
GNSS Object	GNSS Data and Qualifier
Output Data	**Description**
Spatial Attitude	Final Spatial Attitude

10.25.4 JSON Metadata

https://schemas.mpai.community/CAV1/V1.1/AIMs/SpatialAttitudeGeneration.json

10.26 Traffic Obstacle Avoidance

10.26.1 Functions

The Traffic Obstacle Avoidance (CAV-TOA) AIM:

Receives	*Full Environment Descriptors*	From Full Environment Description.
	Trajectory	From Motion Selection Planning.
	AMS Data	From AMS Memory.
	Alert	Alert message form AMS.
	AMS-MAS Message	Message from MAS.
Produces	*Full Environment Descriptors*	To Full Environment Description.
	Road State	To Full Environment Description.
	CAV State	To Full Environment Description.
	AMS-MAS Message	Message to MAS.
	Alert	To AMS Decision Recording.

10.26.2Reference Architecture

Figure 1 depicts the Reference Architecture of the Traffic Obstacle Avoidance (CAV-TOA) AIM.

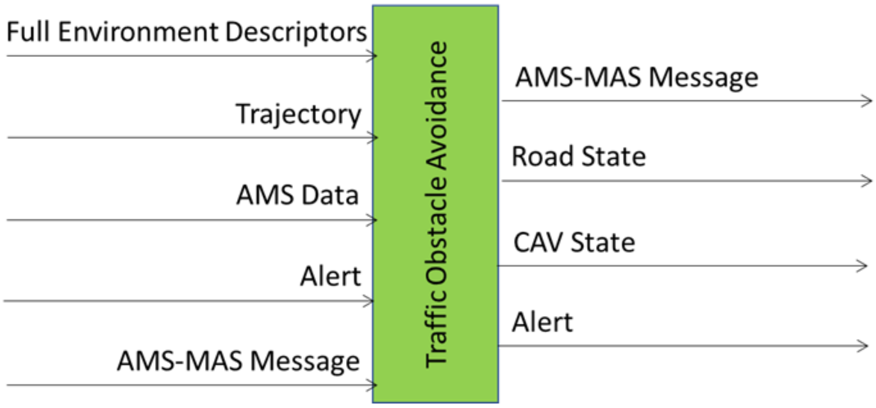

Figure 1 – The Traffic Obstacle Avoidance (CAV-TOA) AIM

10.26.3I/O Data

Table 1 specifies the Input and Output Data of the Traffic Obstacle Avoidance (CAV-TOA) AIM.

Table 1 – I/O Data of the Traffic Obstacle Avoidance (CAV-TOA) AIM

Input	Description
Full Environment Descriptors	Provided by CAV-FED AIM.
Trajectory	From Motion Selection Planning.
AMS Data	CAV Data from AMS Memory.
Alert	From ESS.
AMS-MAS Message	Message from MAS.
Output	**Description**
Full Environment Descriptors	FED updated based on AMS-MAS-Message received from MAS.
AMS-MAS Message	Message to MAS.
Road State	From AMS-MAS Message response.
CAV State	From AMS-MAS Message response.
Alert	To AMS Memory

10.26.4JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/TrafficObstacleAvoidance.json

10.27 Trajectory Planning and Decision

10.27.1 Functions

The Trajectory Planning and Decision (CAV-TPD) AIM:

Receives	*AMS Data*	Data collected by AMS Memory.
	Alert	Alert message form ESS.
	AMS-MAS Message	Response from MAS.
	Full Environment Descriptors	From Full Environment Description.
	Route	From Route Selection Planning.
Produces	*AMS-MAS Message*	Message to MAS.
	CAV State	To Full Environment Description and MAS.
	Road State	To Full Environment Description and MAS.
	Alert	To AMS Memory.

10.27.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the Trajectory Planning and Decision (CAV-TPD) AIM.

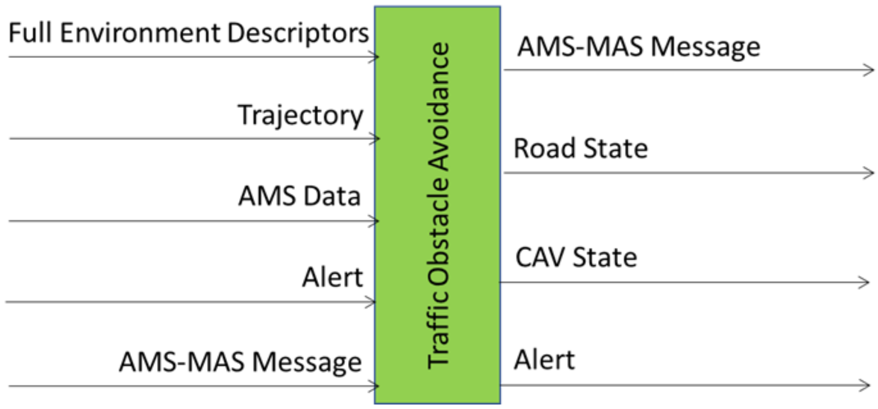

Figure 1 – The Trajectory Planning and Decision (CAV-TPD) AIM

10.27.3 I/O Data

Table 1 specifies the Input and Output Data of the Trajectory Planning and Decision (CAV-TPD) AIM.

Table 1 – I/O Data of the Trajectory Planning and Decision (CAV-TPD) AIM

Input	Description
AMS Data	Data collected by AMS Memory.
Alert	Alert message form ESS.
AMS-MAS Message	Response from MAS.

Full Environment Descriptors	From Full Environment Description.
Route	From Route Selection Planning.
Output	**Description**
AMS-MAS Message	Message to MAS.
CAV State	Update from AMS-MAS Message.
Road State	Update from AMS-MAS Message.
Alert	Alert for storage in AMS Memory

10.27.4SubAIMs

Trajectory Planning and Decision (CAV-TPD) AIM is a Composite AIM whose reference Model is depicted in Figure 2.

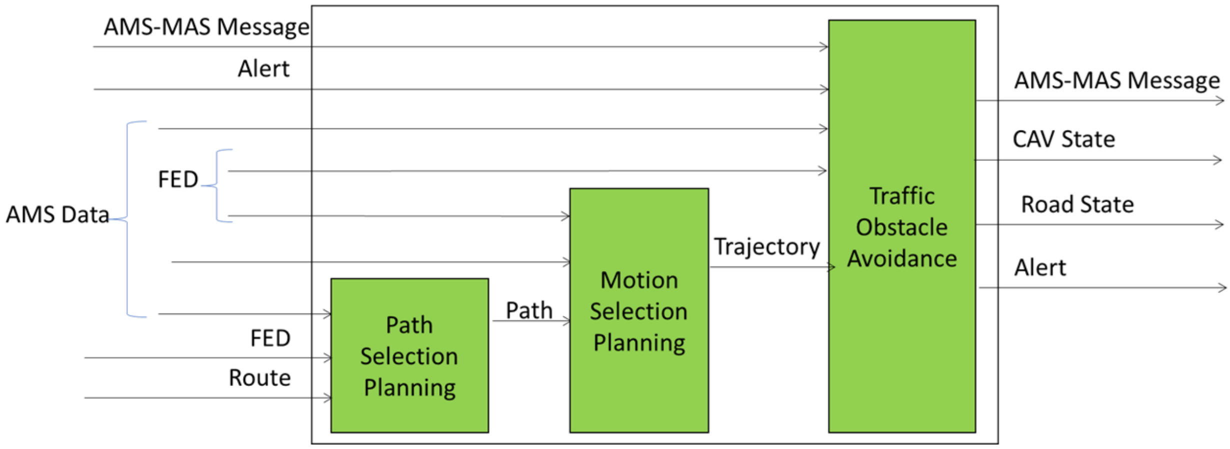

Figure 2 – Reference Model of Trajectory Planning and Decision (CAV-TPD) Composite AIM

The AIMs composing the Trajectory Planning and Decision (CAV-TPD) Composite AIM are:

Composite AIM	AIM	Name	JSON
CAV-TPD	CAV-TPD	Trajectory Planning and Decision	X
	CAV-PSP	Path Selection Planning	X
	CAV-MSP	Motion Selection Planning	X
	CAV-TOA	Traffic Obstacle Avoidance	X

10.27.5JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/TrajectoryPlanningAndDecision.json

10.28 AMS-MAS Message Interpretation

10.28.1Functions

AMS Command Interpretation (CAV-ACI):

Receives	*AMS-MAS Message*	From AMS Command issuer
Produces	*Brake Command*	To mechanical subsystem
	Motor Command	To mechanical subsystem

Wheel Command To mechanical subsystem

10.28.2Reference Model

Figure 1 depicts the Reference Model of the MAS Spatial Attitude Generation AIM.

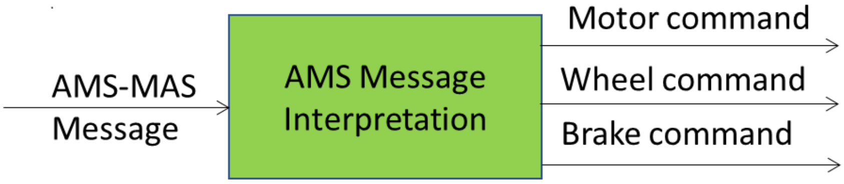

Figure 1 – The MAS Spatial Attitude Generation AIM Reference Model

10.28.3I/O Data

Table 1 specifies the Input and Output Data of the MAS Spatial Attitude Generation AIM.

Table 1 – I/O Data of the MAS Spatial Attitude Generation AIM

Input	**Description**
AMS-MAS Message	Message from the AMS.
Output	**Description**
Brake Command	Conversion of AMS-MAS Message to specific Brake Command to mechanical subsystem.
Motor Command	Conversion of AMS-MAS Message to specific Motor Command to mechanical subsystem.
Wheel Command	Conversion of AMS-MAS Message to specific Wheel Command to mechanical subsystem.

10.28.4 JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/AMSMASMessageInterpretation.json

10.29 Ice Condition Analysis

10.29.1 Functions

The Ice Condition Analysis (CAV-ICA) AIM:

Receives	*Weather Data*	From internal sensing devices.
	Brake Response	From mechanical subsystem
	Motor Response	From mechanical subsystem
	Wheel Response	From mechanical subsystem
Produces	*Weather Data*	With added Ice Conditions

10.29.2 Reference Model

Figure 1 depicts the Reference Architecture of the Ice Condition Analysis (CAV-ICA) AIM.

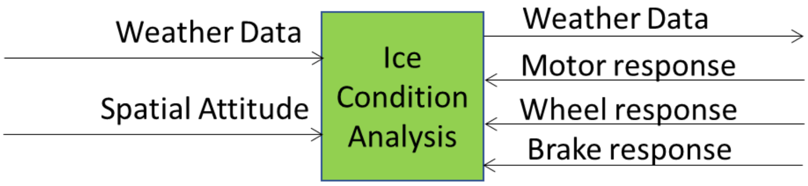

Figure 1 – The Ice Condition Analysis (CAV-ICA) AIM Reference Model

10.29.3 I/O Data

Table 1 specifies the Input and Output Data of the MAS Spatial Attitude Generation AIM.

Table 1 – I/O Data of the Ice Condition Analysis (CAV-ICA) AIM

Input	**Description**
Weather Data	Produced by sensors.
Spatial Attitude	Available at MAS.
Motor Response	From mechanical subsystem
Wheel Response	From mechanical subsystem
Brake Response	From mechanical subsystem
Output	**Description**
Weather Data	Augmented with Ice Conditions.

10.29.4JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/IceConditionAnalysis.json

10.30 MAS Response Analysis

10.30.1 Functions

The MAS Response Analysis (CAV-MRA) AIM:

Receives	*Brake Response*	From mechanical subsystem
	Motor Response	From mechanical subsystem
	Wheel Response	From mechanical subsystem
Produces	*AMS-MAS Message*	To AMS

10.30.2 Reference Architecture

Figure 1 depicts the Reference Architecture of the MAS Response Analysis AIM.

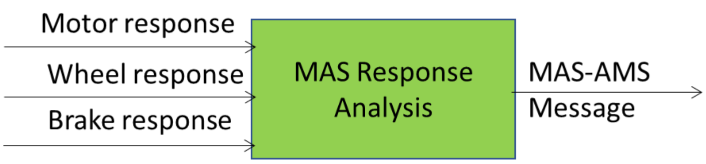

Figure 1 – The MAS Response Analysis AIM

10.30.3 I/O Data

Table 1 specifies the Input and Output Data of the MAS Response Analysis AIM.

Table 1 – I/O Data of the MAS Response Analysis AIM

Input	Description
Brake Response	From Brake in mechanical subsystem.
Motor Response	From Motor in mechanical subsystem.
Wheel Response	From Wheel in mechanical subsystem.
Output	**Description**
AMS-MAS Message	Message to AMS with analysis of response of mechanical subsystem,.

10.30.45 JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/MASResponseAnalysis.json

10.31 MAS Response Analysis

10.31.1 Functions

The MAS Response Analysis (CAV-MRA) AIM:

Receives	*Brake Response*	From mechanical subsystem
	Motor Response	From mechanical subsystem
	Wheel Response	From mechanical subsystem
Produces	*AMS-MAS Message*	To AMS

10.31.2Reference Architecture

Figure 1 depicts the Reference Architecture of the MAS Response Analysis AIM.

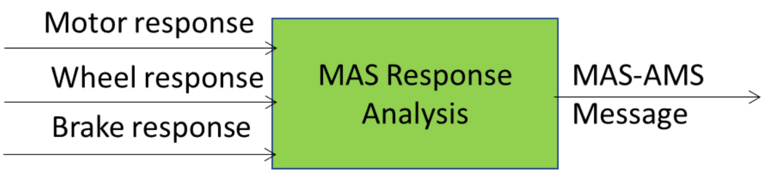

Figure 1 – The MAS Response Analysis AIM

10.31.3I/O Data

Table 1 specifies the Input and Output Data of the MAS Response Analysis AIM.

Table 1 – I/O Data of the MAS Response Analysis AIM

Input	Description
Brake Response	From Brake in mechanical subsystem.
Motor Response	From Motor in mechanical subsystem.
Wheel Response	From Wheel in mechanical subsystem.
Output	**Description**
AMS-MAS Message	Message to AMS with analysis of response of mechanical subsystem,.

10.31.4JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/MASResponseAnalysis.json

10.32 MAS Spatial Attitude Generation

10.32.1 Functions

The MAS Spatial Attitude Generation (CAV-MSA) AIM:

Receives	*Spatial Data*	From Odometer Data, Speedometer Data, Accelerometer Data, and Inclinometer Data.
Produces	*Spatial Attitude*	Initial estimate of CAV's Spatial Attitude using information available at MAS.

10.32.2Reference Architecture

Figure 1 depicts the Reference Architecture of the MAS Spatial Attitude Generation AIM.

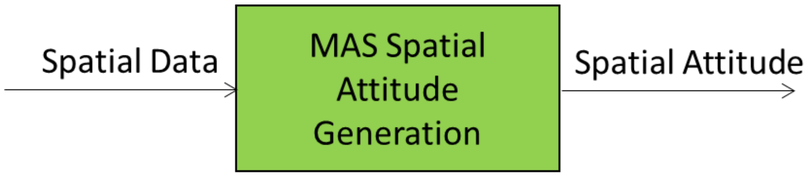

Figure 1 – The MAS Spatial Attitude Generation AIM

10.32.3I/O Data

Table 1 specifies the Input and Output Data of the MAS Spatial Attitude Generation AIM.

Table 1 – I/O Data of the MAS Spatial Attitude Generation AIM

Input	Description
Spatial Data	Set of CAV internal spatial information.
Output	**Description**
Spatial Attitude	Best estimate of CAV Spatial Attitude using data available at MAS.

10.32.4JSON Metadata

https://schemas.mpai.community/CAV2/V1.0/AIMs/MASSpatialAttitudeGeneration.json

11 Data Types

Table 1 lists the Data Types used by CAV-TEC V1.0 organised by CAV Subsystems. Each entry includes a link to the relevant specifications. Note that several Data Types are specified by other MPAI Technical Specifications than CAV-TEC: MPAI-CAE, MPAI-MMC, MPAI-OSD, MPAI-PAF, and MPAI-TFA.

Table 1 - Data Types organised by CAV Subsystems

Human-CAV Interaction	Environment Sensing Subsystem	Autonomous Motion Subsystem	Motion Actuation Subsystem
HCI	**ESS**	**AMS**	**MAS**
3D Model Object	Alert	AMS-MAS Message	Brake Command
AMS-HCI Message	Basic Environment Descriptors	AMS Data	Brake Response
Annotation	Bounding Box	CAV Identifier	Motor Command
Audio Object	Coordinates	CAV State	Motor Response
AV Scene Descriptors	GNSS Object	Ego-Remote AMS Message	Spatial Data
Body Descriptors	LiDAR Object	Full Environment Descriptors	Weather Data
Ego-Remote HCI Message	LiDAR Scene Descriptors	Road Attributes	Wheel Command
Face Descriptors	Location	Route	Wheel Response
Instance Identifier	Offline Map Object		
Meaning	Offline Map Scene Descriptors		
Orientation	Path		
Perceptible Entity	Point of View		
Personal Preferences	Position		
Personal Profile	RADAR Object		
Personal Status	RADAR Scene Descriptors		
Portable Avatar	Road State		
Speech Object	Space-Time		
Text Object	Spatial Attitude		
Visual Object	Time		
	Traffic Rules		
	Traffic Sign Objects		
	Trajectory		
	Ultrasound Object		
	Ultrasound Scene Descriptors		

Data Types are sequentially specified in by Subsystems.

11.1 3D Model Object

11.1.1 Definition

A Data Type including a collection of Basic 3D Model Objects.

A 3D Model Object can have a hierarchical structure where 3D Model Objects contain Basic 3D Model Objects and 3D Model Objects.

11.1.2 Functional Requirements

A 3D Model Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the 3D Model Object.
3. Space-Time information of the 3D Model Object.
4. Basic 3D Model Object and 3D Model Objects included in the 3D Model Objects.
5. Annotation data set including:
    1. Annotations
    2. Space-Times of the Annotations.
    3. Rights to perform Actions on the 3D Model Object.
6. The Rights that may be exercised on the 3D Model Object.

Note that.

1. An 3D Model Object that does not include Sub-Scenes and only one Basic 3D Model Object is a Basic 3D Model Object.
2. The Space-Time information of a Basic 3D Model Object and 3D Model Object included in an 3D Model Object may be superseded by the Space-Time information of the 3D Model Object containing them.

11.1.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/3DModelObject.json

11.1.4 Semantics

Label	Size	Description
Header	N1 Bytes	3D Model Object Header
– Standard-3D ModelObject	9 Bytes	The characters “OSD-3DO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
3DModelObjectID	N5 Bytes	Identifier of the 3D Model Object.
3DModelObjectSpaceTime	N6 Bytes	Space-Time of 3D Model Object.
Basic3DModelObjectCount	N7 Bytes	Set of Parent 3D Model Objects.

Basic3DModelObjects[]	N8 Bytes	Set of Basic 3D Model Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic 3D Model Object in the 3D Model Object.
- Basic3DModelObject	N10 Bytes	A Basic 3D Model Object in the 3D Model Object.
3DModelObjectCount	N11 Bytes	Number of 3D Model Objects.
3DModelObjects[]	N12 Bytes	Set of 3D Model Objects.
- SpaceTime	N13 Bytes	Space Time of an 3D Model Object in the 3D Model Object.
- 3DModelObject	N14 Bytes	A 3D Model Object in the 3D Model Object
Annotations[]	N15 Bytes	Set of 3D Model Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.1.5 Conformance Testing

A Data instance Conforms with 3D Model Object (OSD-3DO) V1.3 if:

1. The Data validates against the 3D Model Object's JSON Schema.
2. All Data in the 3D Model Object's JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.2 AMS-HCI Message

11.2.1 Definition

AMS-HCI Messages are exchanged between the Ego CAV's Human-CAV Interaction Subsystem (HCI) and the Autonomous Motion Subsystem (AMS).

11.2.2 Functional Requirements

HCI sends Messages to AMS requesting to:

1. Send
    1. A Route connecting the current and the destination Poses, possibly including intermediate Poses and desired Times. Poses refer to a Offline Map.
    2. The selected Route as a result of an exchange of Routes between HCI and AMS.
    3. One of the following Commands:
        1. Execute a Route.
        2. Suspend a Route.
        3. Resume a Route.
        4. Change a Route.

5. Stop a Route.
2. Request to stream the M-Location corresponding to the human-specified U-Location via Point of View.

AMS sends AMS-HCI Messages to HCI to:

1. List of Route options in response to an HCI-AMS Message requesting it.
2. Information, such as:
    1. Road State.
    2. CAV State.
    3. Failure ID of equipment.

The message set is extensible.

11.2.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/AMSHCIMessage.json

11.2.4 Semantics

Label	Size	Description
Header	N1 Bytes	AMS-HCI Message Header
- Standard-AMSHCIMessage	9 Bytes	The characters "CAV-AHM-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
AMSHCIMessageID	N4 Bytes	Identifier of AMS-HCI Message.
HCIMessage	N4 Bytes	Data in HCI to AMS Message.
- RequestedRoutes[]	N5 Bytes	Requested Route with Stops
- Route	N6 Bytes	A Requested Route
- OfflineMapID	N7 Bytes	Reference Offline Map ID
- SelectedRouteID	N9 Bytes	ID of selected Route
- RouteCommands	N10 Bytes	"Execute", "Suspend", "Resume", "Change", "Stop"
- StreamPointOfView	N11 Bytes	Coordinates of Point from where to watch environment.
- ULocation	N12 Bytes	U-Location that passenger wishes to see from Point of View
AMSMessage	N13 Bytes	Data in AMS to HCI Message.
- RouteList	N14 Bytes	List of Routes
- FailureID	N15 Bytes	CAV Failure ID one 0f Battery low, Mechanics.
- CAVState	N16 Bytes	CAV State information.

- RoadState	N17 Bytes	Road State information.
DescrMetadata	N18 Bytes	ID of AMS Messages

11.3 Annotation

11.3.1 Definition

Annotation is Data attached to an Object or a Scene. As opposed to Qualifier that describes intrinsic properties of an Object, an Annotation is spatially and temporally local and changeable.

11.3.2 Functional Requirements

Elements of an Annotation are:

1. M-Instance ID
2. Annotation ID
3. Annotation Space-Time
4. Annotation Data
    1. JSON Text Objects
    2. Annotation Space-Time in Object or Scene
    3. Permitted Actions on Annotated Data

Annotation Data is text containing the JSON code conforming to the JSON Schema of the Item intended as Annotation. Examples of such Items are Perceptible Entities, Intention, Meaning, and Personal Status and Its components.

11.3.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/Annotation.json

11.3.4 Semantics

Label	Size	Description
Header	N1 Bytes	Annotation Header
- Standard-Annotation	9 Bytes	The characters "OSD-ANN-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
MInstanceID	N4 Bytes	The Virtual Space whose Object or Scene contains Annotations.
AnnotationID	N5 Bytes	Identifier of Annotation.
Annotation[]	N6 Bytes	The actual Annotation.
- AnnotationJSONText	N7 Bytes	Text of the JSON representing the Data Type used in the Annotation.
- AnnotationSpaceTime	N8 Bytes	Where/when Annotation is attached.
- ProcessActions[]	N9 Bytes	What is possible to do with the Annotation

- ProcessActionID	N10 Bytes	List of possible Process Actions
DescrMetadata	N11 Bytes	Descriptive Metadata

11.3.5 Conformance Testing

A Data instance Conforms with MPAI-OSD V1.3 Annotation (OSD-ANN) if:

1. The Data validates against the Annotation's JSON Schema.
2. All Data in the Annotation's JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers if present.

11.4 Audio Object

11.4.1 Definition

A Data Type including a collection of Basic Audio Objects.

An Audio Object can have a hierarchical structure where Audio Objects contain Basic Audio Objects and Audio Objects.

11.4.2 Functional Requirements

An Audio Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the Audio Object.
3. Space-Time information of the Audio Object.
4. Basic Audio Object and Audio Objects included in the Audio Objects.
5. Annotation data set including:
    1. Annotations
    2. Space-Times of the Annotations.
    3. Rights to perform Actions on the Audio Object.
6. The Rights that may be exercised on the Audio Object.

Note that.

1. An Audio Object that does not include Sub-Scenes and only one Basic Audio Object is a Basic Audio Object.
2. The Space-Time information of a Basic Audio Object, Audio Object included in an Audio Object may be superseded by the Space-Time information of the Audio Object containing it.

11.4.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/AudioObject.json

11.4.4 Semantics

Label	Size	Description

Header	N1 Bytes	Audio Object Header
– Standard-AudioObject	9 Bytes	The characters “OSD-AUO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
AudioObjectID	N5 Bytes	Identifier of the Audio Object.
AudioObjectSpaceTime	N6 Bytes	Space-Time of Audio Object.
BasicAudioObjectCount	N7 Bytes	Set of Parent Audio Objects.
BasicAudioObjects[]	N8 Bytes	Set of Basic Audio Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic Audio Object in the Audio Object.
- BasicAudioObject	N10 Bytes	A Basic Audio Object in the Audio Object.
AudioObjectCount	N11 Bytes	Number of Audio Objects.
AudioObjects[]	N12 Bytes	Set of Audio Objects.
- SpaceTime	N13 Bytes	Space Time of an Audio Object in the Audio Object.
- AudioObject	N14 Bytes	An Audio Object in the Audio Object
Annotations[]	N14 Bytes	Set of Audio Object Annotation.
– Annotation	N15 Bytes	An Annotation.
– AnnotationSpaceTime	N15 Bytes	Where Annotation is attached and when it will be active.
– Rights	N16 Bytes	Actions that may be performed on the Annotation
Rights	N17 Bytes	Actions that may be performed on the Object.
DescrMetadata	N17 Bytes	Descriptive Metadata

11.4.5 Conformance Testing

A Data instance Conforms with Audio Object (OSD-AUO) V1.3 if:

1. The Data validates against the Audio Object’s JSON Schema.
2. All Data in the Audio Object’s JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.5 Audio-Visual Scene Descriptors

11.5.1 Definition

A Data Type including the Audio-Visual Scene's Objects and Sub-Scenes and their arrangement in the Scene.

11.5.2 Functional Requirements

Audio-Visual Scene Descriptors includes Scenes in addition to Objects.

11.5.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/AudioVisualSceneDescriptors.json

11.5.4 Semantics

Label	Size	Description
Header	N1 Bytes	Audio-Visual Scene Descriptors Header
- Standard-AVSceneDescriptors	9 Bytes	The characters “OSD-AVS-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
AVBasicSceneDescriptorsID	N5 Bytes	Identifier of the AV Object.
ObjectCount	N6 Bytes	Number of Objects in Scene
AVSceneSpaceTime	N7 Bytes	Data about Space and Time
SpeechObjects[]	N8 Bytes	Set of Speech Objects
- SpeechObject	N9 Bytes	Speech Object
- SpeechObjectSpaceTime	N10 Bytes	Space-Time of Speech Object
AudioObjects[]	N11 Bytes	Set of Audio Objects
- AudioObject	N12 Bytes	ID of Audio Object
- AudioObjectSpaceTime	N13 Bytes	Space-Time of Audio Object
VisualObjects[]	N14 Bytes	Set of Visual Objects
- VisualObjectID	N15 Bytes	ID of Visual Object
- VisualObjectSpaceTime	N16 Bytes	Space-Time of Visual Object
AudioVisualObjects[]	N17 Bytes	Set of Audio-Visual Objects
- AudioVisualObjectID	N18 Bytes	ID of Audio-Visual Object
- AudioObjectSpaceTime	N19 Bytes	Space-Time of Audio-Visual Object
SubSceneCount	N20 Bytes	Number of Sub-Scenes in Scene
SpeechSubScenes[]	N21 Bytes	Set of Speech Objects
- SpeechSubScene	N22 Bytes	Speech SubScene
- SpeechSubSceneSpaceTime	N23 Bytes	Space-Time of Speech SubScene
AudioSubScenes[]	N24 Bytes	Set of Audio SubScenes
- AudioSubScene	N25 Bytes	ID of Audio SubScene
- AudioSubSceneSpaceTime	N26 Bytes	Space-Time of Audio SubScene
VisualSubScenes[]	N27 Bytes	Set of Visual SubScenes
- VisualSubSceneID	N28 Bytes	ID of Visual SubScene
- VisualSubSceneSpaceTime	N29 Bytes	Space-Time of Visual SubScene
AudioVisualSubScenes[]	N30 Bytes	Set of Audio-Visual SubScenes
- AudioVisualSubSceneID	N31 Bytes	ID of Audio-Visual SubScene

- AudioSubSceneSpaceTime	N31 Bytes	Space-Time of Audio-Visual SubScene
DescrMetadata	N33 Bytes	Descriptive Metadata

11.5.5 Conformance Testing

A Data instance Conforms with Audio-Visual Scene Descriptors (OSD-AVS) V1.3 if:

1. The Data validates against the Audio-Visual Scene Descriptors' JSON Schema.
2. All Data in the Audio-Visual Scene Descriptors' JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers if present.

11.6 Body Descriptors

11.6.1 Definition

Body Descriptors is a Data Type digitally representing a human or a humanoid.

Gesture Descriptors is a Data Type representing the subset of Body Descriptors selected by an application to convey Gesture information.

11.6.2 Functional Requirements

Body Descriptors should enable the representation of the joints of a body.

11.6.3 Syntax

Syntax is given by Reference. The Body Descriptors XML Syntax is given by: https://www.web3d.org/x3d/content/examples/X3dResources.html

11.6.4 Semantics

The semantics of Body Descriptors is provided by https://www.web3d.org/content/hanim-architecture-v2.

11.6.5 Conformance Testing

A Data instance Conforms with Body Descriptors (PAF-BDD) V1.4 if the Data instance validates against the Body Descriptors XML Schema.

11.7 Ego-Remote HCI Message

11.7.1 Definition

Message exchanged between the Ego CAV's and the Human-CAV Interaction Subsystems (HCI) of another CAV or CAV-Aware entity (e.g., Roadside Unit, a Store and Forward entity) "Remote HCI".

11.7.2 Functional Requirements

Ego HCI may:

1. Exchange messages with a Remote HCI.
2. At a passenger's prompt, request a Remote HCI to send the digital representation (M-Location) of the audio-visual scene of a specific U-Location.
3. Respond or not to the request from a Remote HCI to send the digital representation (M-Location) of the audio-visual scene of a specific U-Location by selecting the Level of Detail for transmitting the requested M-Location with the available bandwidth.
4. Respond to a request to move the CAV to a specified Wai Point and stop assuming a specified Point of View.

11.7.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/EgoRemoteHCIMessage.json

11.7.4 Semantics

Label	Size	Description
Header	N1 Bytes	Ego-Remote HCI Message Header
- Standard - EgoRemoteHCIMessage	9 Bytes	The characters "MMM-ERH-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
EgoRemoteHCIMessageID	N4 Bytes	Identifier of Ego-Remote HCI Message.
EgoRemoteHCIMessage	N5 Bytes	Data of Ego-Remote-HCI Message.
- Request	N6 Bytes	Data sent on Ego CAV's initiative.
- MLocationRequest	N7 Bytes	M-Location corresponding to a U-Location.
- GenericMessage	N8 Bytes	Any message.
- Response	N9 Bytes	Data sent in response to a request or message.
- FullEvironmentDescriptors	N10 Bytes	Full Environment Descriptors.
- GenericMessage	N11 Bytes	Response to Message.
DescrMetadata	N12 Bytes	Descriptive Metadata.

11.8 Face Descriptors

11.8.1 Definition

Face Descriptors is a Data Type representing the features of the Face of an Entity.

11.8.2 Functional Requirements

The Face Descriptors represent the effect of the motion of the muscles of a human face.

The Face Descriptors Syntax represents the Actions Units of the Facial Action Coding System (FACS) originally developed by Carl-Herman Hjortsjö, adopted by Paul Ekman and Wallace V. Friesen (1978) and updated by Ekman, Friesen, and Joseph C. Hager (2002).

11.8.3 Syntax

https://schemas.mpai.community/PAF/V1.4/data/FaceDescriptors.json

11.8.4 Semantics

Header		N1 Bytes	Orientation FaceDescriptors
- Standard-FaceDescriptors		9 Bytes	The characters “OSD-FCD-V”
- Version		N2 Bytes	Major version – 1 or 2 characters
- Dot-separator		1 Byte	The character “.”
- Subversion		N3 Bytes	Minor version – 1 or 2 characters
FaceDescriptorsID		N4 Bytes	Identifier of Face Descriptors.
AU	**Description**	N5 Bytes	Facial muscle generating the Action
1	Inner Brow Raiser	1 Byte	Frontalis, pars medialis
2	Outer Brow Raiser	1 Byte	Frontalis, pars lateralis
4	Brow Lowerer	1 Byte	Corrugator supercilii, Depressor supercilii
5	Upper Lid Raiser	1 Byte	Levator palpebrae superioris
6	Cheek Raiser	1 Byte	Orbicularis oculi, pars orbitalis
7	Lid Tightener	1 Byte	Orbicularis oculi, pars palpebralis
9	Nose Wrinkler	1 Byte	Levator labii superioris alaquae nasi
10	Upper Lip Raiser	1 Byte	Levator labii superioris
11	Nasolabial Deepener	1 Byte	Zygomaticus minor
12	Lip Corner Puller	1 Byte	Zygomaticus major
13	Cheek Puffer	1 Byte	Levator anguli oris (a.k.a. Caninus)
14	Dimpler	1 Byte	Buccinator
15	Lip Corner Depressor	1 Byte	Depressor anguli oris (a.k.a. Triangularis)
16	Lower Lip Depressor	1 Byte	Depressor labii inferioris
17	Chin Raiser	1 Byte	Mentalis
18	Lip Puckerer	1 Byte	Incisivii labii superioris and Incisivii labii inferioris
20	Lip stretcher	1 Byte	Risorius with platysma
22	Lip Funneler	1 Byte	Orbicularis oris
23	Lip Tightener	1 Byte	Orbicularis oris
24	Lip Pressor	1 Byte	Orbicularis oris
25	Lips part	1 Byte	Depressor labii inferioris or relaxation of Mentalis, or Orbicularis oris
26	Jaw Drop	1 Byte	Masseter, relaxed Temporalis and internal Pterygoid
27	Mouth Stretch	1 Byte	Pterygoids, Digastric
28	Lip Suck	1 Byte	Orbicularis oris
41	Lid droop	1 Byte	Relaxation of Levator palpebrae superioris
42	Slit	1 Byte	Orbicularis oculi

43	Eyes Closed	1 Byte	Relaxation of Levator palpebrae superioris; Orbicularis oculi, pars palpebralis
44	Squint	1 Byte	Orbicularis oculi, pars palpebralis
45	Blink	1 Byte	Relaxation of Levator palpebrae superioris; Orbicularis oculi, pars palpebralis
46	Wink	1 Byte	Relaxation of Levator palpebrae superioris; Orbicularis oculi, pars palpebralis
61	Eyes turn left	1 Byte	Lateral rectus, medial rectus
62	Eyes turn right	1 Byte	Lateral rectus, medial rectus
63	Eyes up	1 Byte	Superior rectus, Inferior oblique
64	Eyes down	1 Byte	Inferior rectus, Superior oblique

11.8.5 Conformance Testing

A Data instance Conforms with Face Descriptors (PAF-FCD) V1.4 if:

1. The Data validates against the Face Descriptors' JSON Schema.
2. All Data in the Face Descriptors' JSON Schema
    1. Have the specified type.
    2. Validate against their JSON Schemas.

11.8.6 Mapping of AUs to Personal Status (Informative)

MPAI has defined a set of Cognitive States, Emotions, and Social Attitudes included in Personal Status. The Table below offers an informative mapping of some elements of Personal Status to Action Units (from 1).

Personal Status	Cognitive State	Emotion	Prototypical (and variant AUs)
Happy		17	12, 25 [6 (51%)]
Sad		32	4, 15 [1 (60%), 6 (50%), 11 (26%), 17 (67%)]
Fearful		13	1, 4, 20, 25 [2 (57%), 5 (63%), 26 (33%)]
Angry		2	4, 7, 24 [10 (26%), 17 (52%), 23 (29%)]
Surprised	18		1, 2, 25, 26 [5 (66%)]
Disgusted		11	9, 10, 17 [4 (31%), 24 (26%)]

This Table was obtained through a series of experiments with human subjects. AUs used by a subset of the subjects are shown in brackets with the percentage of the subjects using this less common AU in parentheses.

[1] Compound facial expressions of emotion | PNAS

11.9 Instance Identifier

11.9.1 Definition

A Data Type associating a string (Identifier) with an element of a set of entities – Speech, Objects, Visual Objects, User IDs etc. – belonging to some levels in a hierarchical classification (taxonomy).

11.9.2 Functional Requirements

Instance Identifier includes:

1. ID of Virtual Space (M-Instance)
2. Instance Label
3. Confidence level of the association between Instance Label and Instance.
4. Taxonomy
5. Confidence level of the association between Taxonomy and the Instance.

11.9.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/InstanceIdentifier.json

11.9.4 Semantics

Label	Size	Description
Header	N1 Bytes	Instance Identifier Header
– Standard-InstanceIdentifier	9 Bytes	The characters "OSD-IID-V"
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character "."
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance
InstanceID	N5 Bytes	Identifier of Instance.
InstanceSpaceTime	N6 Bytes	Data about Space-Time
InstanceIdentifierData	N7 Bytes	Data set of Instance Identifier.
InstanceLabel	N8 Bytes	Instance identified by Instance Identifier.
LabelConfidenceLevel	N9 Bytes	Confidence of Instance Label and Instance association.
TaxonomyLabel	N10 Bytes	Taxonomy Instance Identifier belongs to.
TaxonomyConfidenceLevel	N11 Bytes	Confidence of Taxonomy Label .
TaxonomyDataLength	N12 Bytes	Number of Bytes
TaxonomyDataURI	N13 Bytes	URI of Taxonomy.
DescrMetadata	N14 Bytes	Descriptive Metadata

11.9.5 Conformance Testing

A Data instance Conforms with Instance Identifier (OSD-IID) V1.3 if:

1. The Data validates against the Instance Identifier's JSON Schema.
2. All Data in the Instance Identifier's JSON Schema

1. Have the specified type
2. Validate against their JSON Schemas
3. Conform with their Data Qualifiers if present.

11.10 Meaning

11.10.1Definition

A Data Type representing the syntactic and semantic information of an input text. Meaning is synonym of Text Descriptors.

11.10.2Functional Requirements

Meaning is used to extract information from text to help the Entity Dialogue Processing AIM to produce a response.

11.10.3Syntax

https://schemas.mpai.community/MMC/V2.3/data/Meaning.json

11.10.4Semantics

Label	Size	Description
Header	N1 Bytes	Meaning Header
- Standard-Meaning	9 Bytes	The characters “MMC-TXD-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
MeaningID	N5 Bytes	Identifier of Meaning.
Meaning	N6 Bytes	Data set of Meaning
- POS_tagging	N7 Bytes	Results of POS (Part of Speech, e.g., noun, verb, etc.) tagging including information on the question’s POS tagging set and tagged results.
- NE_tagging	N8 Bytes	Results of NE (Named Entity e.g., Person, Organisation, Fruit, etc.) tagging results including information on the question’s tagging set and tagged results.
- Dependency_tagging	N9 Bytes	Results of dependency (structure of the sentence, e.g., subject, object, head of relation, etc.) tagging including information on the question’s dependency tagging set and tagged results.
- SRL_tagging	N10 Bytes	Results of SRL (Semantic Role Labelling) tagging results including information on the question’s SRL tagging set and tagged results. SRL indicates the semantic structure of the sentence such as agent, location, patient role, etc.
DescrMetadata	N11 Bytes	Descriptive Metadata

11.10.5 Conformance Testing

A Data instance Conforms with MPAI-MMC V2.3 Meaning (MMC-MEA) if:

1. The Data validates against the Meaning's JSON Schema.
2. All Data in the Meaning's JSON Schema have the specified type.

11.11 Perceptible Entity

11.11.1Definition

Perceptible Entity is one of

1. Text, Speech, Audio, Visual, 3D Model, and Audio-Visual Object.
2. Speech, Audio, Visual, 3D Model, and Audio-Visual Scene.
3. Audio-Visual Event.

11.11.2Functional Requirements

A Perceptible Entity

1. Inherits the Functional requirements of Objects, Scenes, and Events listed above.
2. May include Rights that are Granted to certain Process to perform certain Actions at certain Times and Locations on the Perceptible Entity.

11.11.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/PerceptibleEntity.json

11.11.4Semantics

Label	Size	Description
Header	N1 Bytes	Perceptible Entity Header
- Standard-PerceptibleEntity	9 Bytes	The characters "OSD-PCE-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
PerceptibleEntityID	N4 Bytes	Identifier of Perceptible Entity.
PerceptibleEntity	N5 Bytes	Anyone of the following Objects, Scenes, or Events.
- TextObject	N6 Bytes	Intended Text Object
- SpeechObject	N7 Bytes	Intended Speech Object
- AudioObject	N8 Bytes	Intended Audio Object
- VisualObject	N9 Bytes	Intended Visual Object
- 3D Model	N10 Bytes	Intended 3D Model Object

- AudioVisualObject	N11 Bytes Intended Audio-Visual Object
- SpeechScene	N12 Bytes Intended Speech Scene
- AudioScene	N13 Bytes Intended Audio Scene
- VisualScene	N14 Bytes Intended Visual Scene
- 3D Model Scene	N15 Bytes Intended 3D Model Scene
- AudioVisualScene	N16 Bytes Intended Audio-Visual Scene
- AudioVisualEvent	N17 Bytes Intended Audio-Visual Event
- RightsID	N18 Bytes Individual Rights ID
- Rights	N19 Bytes Individual Rights
DescrMetadata	N20 Bytes Descriptive Metadata

11.11.5 Conformance Testing

A Data instance Conforms with Perceptible Entity (OSD-PCE) V1.3 if:

1. The Data validates against the Perceptible Entity's JSON Schema.
2. All Data in the Perceptible Entity's JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers if present.

11.12 Personal Preferences

11.12.1 Definition

Personal Preferences is a Data Type that includes passenger-specific preferences that enable a Human-CAV Interaction (HCI) Subsystem to access information that facilitates human-HCI interaction. This is particularly useful when the passenger uses a rented CAV, so that the human preferences can be easily communicated to a new CAV.

11.12.2 Functional Requirements

The data in the Personal Preferences should include:

1. Language
2. Seat position.
3. Mirror position.
4. Display characteristics.
5. Preferred driving style.
6. Preferential routes.
7. Preferred information sources.
8. Preferred entertainment sources
9. …

11.12.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/PersonalPreferences.json

11.12.4 Semantics

Label	Size	Description
Header	N1 Bytes	
- Standard	9 Bytes	The characters "CAV-PPR-V"
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Byte	Minor version – 1 or 2 characters
humanID	N4 Bytes	ID of the human the Personal Profile refers to.
PersonalPreferenceID	N5 Bytes	ID of Personal Profile.
PersonalPreferences	N6 Bytes	Set of Personal Preferences.
- Language	N7 Bytes	Preferred Language.
- Seat position.	N8 Bytes	Preferred seat position.
- Mirror position.	N9 Bytes	Preferred mirror position
- Display characteristics.	N10 Bytes	Preferred display characteristics
- Preferred driving style.	N11 Bytes	Preferred driving style
- Preferential routes	N12 Bytes	Preferred routes.
- Preferred information sources	N13 Bytes	Preferred information sources
- Preferred entertainment sources	N14 Bytes	Preferred entertainment sources
DescrMetadata	N15 Bytes	Descriptive Metadata

11.13 Personal Profile

This specification is shared with the planned *Technical Specification: MPAI-Metaverse Model (MPAI-MMM) – Technologies (MMM-TEC) V1.0*.

11.13.1 Definition

Data identifying and describing a human passenger.

11.13.2 Functional Requirements

Personal Profile includes humanID and First Name, Last Name, Age, Nationality, and Email of the human.

11.13.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/PersonalProfile.json

11.13.4 Semantics

Label	Size	Description
Header	N1 Bytes	Personal Profile Header

- Standard - PersonalProfile	9 Bytes	The characters “CAV-PPF-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Byte	Minor version – 1 or 2 characters
humanID	N5 Bytes	ID of the human the Personal Profile refers to.
PersonalProfileID	N6 Bytes	ID of Personal Profile.
PersonalProfile	N7 Bytes	The number of Bytes composing the Personal Profile.
- First Name	N8 Bytes	The human’s given name
- Last Name	N9 Bytes	The human’s family name
- Age	N10 Bytes	The human’s age
- Gender	N11 Bytes	
- Nationality	N11 Bytes	The human’s country
- Email	N12 Bytes	The human’s address
- Preferred pronoun	N13 Bytes	
- Special Needs		
- Visual		AnyOf Blind, Limited vision, Colour blindness
- Oral		Unable to speak, Bad pronounciation
- Hearing		One of Totally deaf, Partially deaf
- Mobility		
- Arms		Unable to use arms
- Legs		One of Unable to walk, Unable to bend legs
- Cognitive		
- Autistic spectrum		
- Dislexia		
- Low understanding		
DescrMetadata	N13 Bytes	Descriptive Metadata

11.14 Personal Status

11.14.1Definition

A Data Type representing the information internal to an Entity that characterises their behaviour.

11.14.2Functional Requirements

Personal Status is a Data Type composed of three *Factors*:

1. *Emotion* (such as “angry” or “sad”).
2. *Cognitive State* (such as “surprised” or “interested”).
3. *Social Attitude* (such as “polite” or “arrogant”).

Factors are expressed by *Modalities*: Text, Speech, Face, and Gestures. (Other Modalities, such as body posture, may be handled in future MPAI Versions.)

Within a given Modality, the Factors can be analysed and interpreted via various *Descriptors*. For example, when expressed via Speech, the elements may be expressed through combinations of such features as prosody (pitch, rhythm, and volume variations); separable speech effects (such as degrees of voice tension, breathiness, etc.); and vocal gestures (laughs, sobs, etc.).

Each of Emotion, Cognitive State, and Social Attitude Factors is represented by a standard set of labels and associated semantics. For each of these Factors, two tables are provided:

- A *Label Set Table* containing descriptive labels relevant to the Factor in a three-level format:
    - The CATEGORIES column specifies the relevant categories using nouns (e.g., “ANGER”).
    - The GENERAL ADJECTIVAL column gives adjectival labels for general or basic labels within a category (e.g., “angry”).
    - The SPECIFIC ADJECTIVAL column gives more specific (sub-categorised) labels in the relevant category (e.g., “furious”).
- A *Label Semantics Table* providing the semantics for each label in the GENERAL ADJECTIVAL and SPECIFIC ADJECTIVAL columns of the Label Set Table. For example, for “angry” the semantic gloss is “emotion due to perception of physical or emotional damage or threat.”

These sets have been compiled in the interests of basic cooperation and coordination among AIM submitters and vendors complemented by a procedure whereby AIM submitters may propose extended or alternate sets for their purposes.

An Implementer wishing to extend or replace a *Label Set Table* for one of the three Factors is requested to do the following:

1. Create a new Label Set Table where:
    1. Proposed additions are clearly marked (in case of extension).
    2. b. All the elements of the target Factor and levels (up to 3) are listed (in case of replacement).
2. Create a new Label Semantics Table where the semantics of elements of the target Factor is:
    1. Added to the semantics of the existing target Factor (in case of extension).
    2. Provided (in case of replacement).
    The submitted semantics should have a level of detail comparable to the semantics given in the current *Label Semantics Table*.
3. Submit both tables to the MPAI Secretariat (secretariat@mpai.community).

The appropriate MPAI Development Committee will examine the proposed extension or replacement. Only the adequacy of the proposed new tables in terms of clarity and completeness will be considered. In case the new tables are not clear or complete, a revision of the tables will be requested.

The accepted External Factor Set will be identified as proposed by the submitter and reviewed by the appropriate MPAI Committee and posted to the MPAI web site.

The versioning system is based on a name – MPAI for MPAI-generated versions or “organisation name” for the proposing organisation – with a suffix m.n where m indicates the version and n indicated the subversion.

11.14.3Syntax

https://schemas.mpai.community/MMC/V2.3/data/PersonalStatus.json

11.14.4Semantics

Label	Size	Description
Header	N1 Bytes	Personal Status Header
- Standard-PersonalStatus	9 Bytes	The characters “MMC-EPS-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
PersonalStatusID	N5 Bytes	Identifier of Meaning.
PersonalStatusSpaceTime	N6 Bytes	Space-Time info of Personal Status
PersonalStatus	N7 Bytes	Personal Status
- CognitiveState	N8 Bytes	Cognitive State component of Personal Status
- Emotion	N9 Bytes	Emotion component of Personal Status
- SocialAttitude	N10 Bytes	Social Attitude component of Personal Status
DescrMetadata	N11 Bytes	Descriptive Metadata

11.15 Portable Avatar

11.15.1Definition

A Data Type that includes:

1. A set of avatar-related Data: M-Instance ID, Avatar ID, Space-Time, Avatar, Language Selector, Text, Speech Object, Personal Status, and
2. Descriptors of the Audio-Visual Scene where the Avatar is embedded and its Space-Time information.

11.15.2Functional Requirements

Portable Avatar provides the following information:

1. The ID of the Virtual Space
2. The set of Data characterising a speaking avatar.
    1. The M-Instance in which the Avatar is located.
    2. The Space-Time of the Avatar.
    3. The Language Preference of the Avatar.

4. The Text the Avatar is associated with, or which will be converted into a Speech Object.
5. The Speech Model used to synthesise Text.
6. The Speech Object that the Avatar utters.
7. The Personal Status of the Avatar.
8. The Space-Time information of the Avatar embedded in the Audio-Visual Scene.

11.15.3Syntax

https://schemas.mpai.community/PAF/V1.4/data/PortableAvatar.json

11.15.4Semantics

Label	Size	Description
Header	N1 Bytes	The Header of the Portable Avatar Data.
– Standard-PortableAvatar	9 Bytes	The characters “PAF-PAV-V”
– Version	N2 Bytes	Major version
– Dot-separator	1 Byte	The character "."
– Subversion	N3 Byte	Minor version
MInstanceID	N4 Bytes	The ID of the M-Instance.
PortableAvatarID	N5 Bytes	Identifier of the Portable Avatar.
PortableAvatarData	N6 Bytes	Set of Data related to Avatar.
- Avatar	N7 Bytes	Avatar's Model and Face and Body Descriptors.
- PortableAvatarSpaceTime	N8 Bytes	Space-Time of Avatar instance in AV Scene.
- LanguageSelector	N9 Bytes	Avatar's Language Preference.
- TextObject	N10 Bytes	Text associated with Avatar.
- SpeechObject	N11 Bytes	Set of Data related to Speech Object.
- SpeechModel	N12 Bytes	Neural Network Model for Speech Synthesis.
- SpeechObject	N13 Bytes	Speech associated with Avatar.
- PersonalStatus	N14 Bytes	Personal Status of Avatar.
- AudioVisualSceneDescriptors	N15 Bytes	AV Scene Descriptors.
- AudioVisualSceneSpaceTime	N16 Bytes	Space and Time info of AV Scene in M-instance.
DescrMetadata	N17 Bytes	Descriptive Metadata

11.15.5Conformance Testing

A Data instance Conforms with Portable Avatar (PAF-PAV) V1.4 if:

1. JSON Data validate against the Portable Avatar 's JSON Schema.
2. All Data in the Portable Avatar 's JSON Schema
    1. Have the specified type.

2. Validate against their JSON Schemas.
3. Conform with their Data Qualifiers if present.

11.16 Speech Object

11.16.1 *Definition*

A Data Type including a collection of Basic Speech Objects.

A Speech Object can have a hierarchical structure where Speech Objects contain Basic Speech Objects and Speech Objects.

11.16.2 *Functional Requirements*

A Speech Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the Speech Object.
3. Space-Time information of the Speech Object.
4. Basic Speech Object and Speech Objects included in the Speech Objects.
5. Annotation data set including:
   1. Annotations
   2. Space-Times of the Annotations.
   3. Rights to perform Actions on the Speech Object.
6. The Rights that may be exercised on the Speech Object.

Note that.

1. A Speech Object that does not include Sub-Scenes and only one Basic Speech Object is a Basic Speech Object.
2. The Space-Time information of a Basic Speech Object and Speech Object included in a Speech Object may be superseded by the Space-Time information of the Speech Object containing them.

11.16.3 *Syntax*

https://schemas.mpai.community/OSD/V1.3/data/SpeechObject.json

11.16.4 *Semantics*

Label	Size	Description
Header	N1 Bytes	Speech Object Header
– Standard-SpeechObject	9 Bytes	The characters “OSD-SPO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
SpeechObjectID	N5 Bytes	Identifier of the Speech Object.
SpeechObjectSpaceTime	N6 Bytes	Space-Time of Speech Object.

BasicSpeechObjectCount	N7 Bytes	Set of Parent Speech Objects.
BasicSpeechObjects[]	N8 Bytes	Set of Basic Speech Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic Speech Object in the Speech Object.
- BasicSpeechObject	N10 Bytes	A Basic Speech Object in the Speech Object.
SpeechObjectCount	N11 Bytes	Number of Speech Objects.
SpeechObjects[]	N12 Bytes	Set of Speech Objects.
- SpaceTime	N13 Bytes	Space Time of a Speech Object in the Speech Object.
- SpeechObject	N14 Bytes	A Speech Object in the Speech Object
Annotations[]	N15 Bytes	Set of Speech Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.16.5 Conformance Testing

A Data instance Conforms with Speech Object (OSD-SPO) V1.3 if:

1. The Data validates against the Speech Object's JSON Schema.
2. All Data in the Speech Object's JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.17 Text Object

11.17.1 *Definition*

A Data Type including a collection of Basic Text Objects.

A Text Object can have a hierarchical structure where Text Objects contain Basic Text Objects and Text Objects.

11.17.2 *Functional Requirements*

A Text Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the Text Object.
3. Space-Time information of the Text Object.
4. Basic Text Object and Text Objects included in the Text Objects.
5. Annotation data set including:
    1. Annotations
    2. Space-Times of the Annotations.

3. Rights to perform Actions on the Text Object.
6. The Rights that may be exercised on the Text Object.

Note that.

1. A Text Object that does not include Sub-Scenes and only one Basic Text Object is a Basic Text Object.
2. The Space-Time information of a Basic Text Object and Text Object included in a Text Object may be superseded by the Space-Time information of the Text Object containing them.

11.17.3 *Syntax*

https://schemas.mpai.community/OSD/V1.3/data/TextObject.json

11.17.4 Semantics

Label	Size	Description
Header	N1 Bytes	Text Object Header
– Standard-TextObject	9 Bytes	The characters "OSD-TXO-V"
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character "."
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
TextObjectID	N5 Bytes	Identifier of the Text Object.
TextObjectSpaceTime	N6 Bytes	Space-Time of Text Object.
BasicTextObjectCount	N7 Bytes	Set of Parent Text Objects.
BasicTextObjects[]	N8 Bytes	Set of Basic Text Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic Text Object in the Text Object.
- BasicTextObject	N10 Bytes	A Basic Text Object in the Text Object.
TextObjectCount	N11 Bytes	Number of Text Objects.
TextObjects[]	N12 Bytes	Set of Text Objects.
- SpaceTime	N13 Bytes	Space Time of a Text Object in the Text Object.
- TextObject	N14 Bytes	A Text Object in the Text Object
Annotations[]	N15 Bytes	Set of Text Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.17.5 *Conformance Testing*

A Data instance Conforms with Text Object (OSD-TXO) V1.3 if:

1. The Data validates against the Text Object's JSON Schema.
2. All Data in the Text Object's JSON Schema

1. Have the specified type
2. Validate against their JSON Schemas
3. Conform with their Data Qualifiers.

11.18 Visual Object

11.18.1Definition

A Data Type including a collection of Basic Visual Objects.

A Visual Object can have a hierarchical structure where Visual Objects contain Basic Visual Objects and Visual Objects.

11.18.2Functional Requirements

A Visual Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the Visual Object.
3. Space-Time information of the Visual Object.
4. Basic Visual Object and Visual Objects included in the Visual Objects.
5. Annotation data set including:
    1. Annotations
    2. Space-Times of the Annotations.
    3. Rights to perform Actions on the Visual Object.
6. The Rights that may be exercised on the Visual Object.

Note that.

1. A Visual Object that does not include Sub-Scenes and only one Basic Visual Object is a Basic Visual Object.
2. The Space-Time information of a Basic Visual Object and Visual Object included in a Visual Object may be superseded by the Space-Time information of the Visual Object containing them.

11.18.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/VisualObject.json

11.18.4*Semantics*

Label	Size	Description
Header	N1 Bytes	Visual Object Header
– Standard-VisualObject	9 Bytes	The characters “OSD-VIO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
VisualObjectID	N5 Bytes	Identifier of the Visual Object.

VisualObjectSpaceTime	N6 Bytes	Space-Time of Visual Object.
BasicVisualObjectCount	N7 Bytes	Set of Parent Visual Objects.
BasicVisualObjects[]	N8 Bytes	Set of Basic Visual Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic Visual Object in the Visual Object.
- BasicVisualObject	N10 Bytes	A Basic Visual Object in the Visual Object.
VisualObjectCount	N11 Bytes	Number of Visual Objects.
VisualObjects[]	N12 Bytes	Set of Visual Objects.
- SpaceTime	N13 Bytes	Space Time of a Visual Object in the Visual Object.
- VisualObject	N14 Bytes	A Visual Object in the Visual Object
Annotations[]	N15 Bytes	Set of Visual Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.18.5Conformance Testing

A Data instance Conforms with Visual Object (OSD-VIO) V1.3 if:

1. The Data validates against the Visual Object’s JSON Schema.
2. All Data in the Visual Object’s JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.19 Alert

11.19.1 Definition

Data sent by a specific Environment Sensing Technology Scene Description AIM to the AMS's Traffic Obstacle Avoider because it is considered to hold information to be processed urgently, hence before the full Basic Scene Descriptors Data is computed.

11.19.2 Functional Requirements

An Alert includes specific objects that are deemed to require consideration. For example, a Scene Description AIM may immediately send a suddenly unobscured traffic sign to the Traffic Obstacle Avoider for consideration.

The EST Scene Description AIM may attach an Annotation to the Object that includes the semantics of the traffic sign Object. It may also add other numerical or textual information extracted from the Object. An example of Annotation is an element of the Vienna Convention on Road Signs and Signals for which CAV-TEC V1.0 provides the sign semantics.

11.19.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/Alert.json

11.19.4 Semantics

Label	Size	Description
Header	N1 Bytes	Alert Header
- Standard	9 Bytes	The characters "CAV-ALT-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
Objects[]	N4 Bytes	Set of relevant Objects.
- Object	N5 Bytes	An Object of the set.
DescrMetadata	N6 Bytes	Descriptive Metadata.

11.20 Basic Environment Descriptors

11.20.1 Definition

Basic Environment Descriptors (BED) digitally represent the environment traversed by a CAV at a time. BED results from the integration of a subset of CAV-sensed data sensed:

1. Spatial Attitude derived from GNSS and Spatial Data.
2. Audio-Visual Scene Descriptors obtained from the fusion of the Scene Descriptors of available Environment Sensing Technologies (EST). The Visual component of the Audio-Visual Scene Descriptors integrates Visual-LiDAR-RADAR-Ultrasound-Offline Map into a scene that is represented by the Visual Scene Format.

11.20.2 Functional Requirements

BEDs at a given Time:

1. Include:
    1. Scene Descriptors of the Environment (possibly improved by the use of static object information, e.g., from Offline Maps).
    2. Other environment information - Weather Data and CAV's Spatial Attitude - enabling the Autonomous Motion Subsystem (AMS) to operate.
    3. The Full Scene Descriptors of a preceding time provided by the Fulle Environment Description AIM.
2. Describe each Object with the following attributes:

    - Object Identifier.
    - AIM Identifier of the AIM that provided the Data used to represent the Object though Object Data Qualifier.
    - Object dimensionality (2D, 2.5D and 3D), shape, and Format through its Qualifier.

- Parent Object(s).
- Spatial Attitude of Object.
- Relationship with other Objects, e.g., groups of Objects (platoon) deduced from the components of the platoon broadcasting Platooning Information, or from observation of a group of CAVs .
- Accuracy of Object values.

11.20.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/BasicEnvironmentDescriptors.json

11.20.4 Semantics

Label	Size	Description
Header	N1 Bytes	Basic Environment Descriptors Headers
- Standard-BasicEnvironmentDescriptors	9 Bytes	The characters "CAV-BED-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
MInstance	N4 Bytes	Virtual Space where BED is (intended) to be placed.
AVSceneDescriptors	N5 Bytes	Audio-Visual Scene Descriptors of Environment.
WeatherData	N6 Bytes	The various elements of weather.
DescrMetadata	N7 Bytes	Descriptive Metadata.

11.21 Bounding Box

11.21.1Definition

A Data Type representing a rectangle (2D Bounding Box) or right parallelepiped (3D Bounding Box) containing a 2D or 3D Visual Object, respectively.

11.21.2Functional Requirements

The rectangle or right parallelepiped is defined, respectively, by

1. Rectangle (2D): 3 vertices not on a straight line.
2. Right Parallelepiped (3D): 4 vertices not on a plane.

The Visual Object (Content) may fit exactly in the rectangle/parallelepiped and have the same axes of the rectangle/parallelepiped.

Content may be absent.

11.21.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/BoundingBox.json

11.21.4 Semantics

Label	Size	Description
Header	N1 Bytes	Bounding Box Header
- Standard-BoundingBox	9 Bytes	The characters “OSD-BBX-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance
BoundingBoxID	N5 Bytes	Identifier of BoundingBox.
Dimensions	2 Bytes	One of 2D, 3D
VisualDataQualifier	N6 Bytes	Qualifier of Visual Data in the BoundingBox.
DescrMetadata	N7 Bytes	Descriptive Metadata

11.21.5 Conformance Testing

A Data instance Conforms with Bounding Box (OSD-BBX) V1.3 if:

1. The Data validates against the Bounding Box’s JSON Schema.
2. All Data in the Bounding Box’s JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers if present.

11.22 Coordinates

11.22.1 Definition

A set of numbers used to indicate the position of a point in a space.

11.22.2 Functional Requirements

All points in the space shall have a set of numbers representing them.

The coordinate systems supported so far are:

1. Cartesian
2. Spherical
3. Geodesic
4. Cylindrical
5. Toroidal

11.22.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/Coordinates.json

11.22.4 Semantics

Label	Size	Description
Header	N1 Bytes	Time Header
- Standard-Object	9 Bytes	The characters "OSD-CRD-V"
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance
CoordinatesID	N5 Bytes	Identifier of Coordinates.
CoordinateTypes	N6 Bytes	One of Cartesian, Spherical, Geodesic, Cylindrical, Toroidal.
CoordinateData	N7 Bytes	Three numbers
DescrMetadata	N8 Bytes	Descriptive Metadata

11.22.5 Conformance Testing

A Data instance Conforms with Coordinates (OSD-CRD) V1.2 if all the Data:

1. Have the specified type.
2. Validate against the Coordinates' JSON Schema.

11.23 GNSS Object

11.23.1 Definition

GNSS Object refers to

1. GNSS Data from a Global Navigation Satellite System (GNSS) obtained from an integration of a constellation of satellites that transmit position and timing data to GNSS receivers and enable them to determine the receivers' location.
2. GNSS Qualifier specified by MPAI-TFA providing information of Sub-Types, Formats and Attributes.

11.23.2 Functional Requirements

GNSS Data can come from four GNSSs – GPS (US), GLONASS (RU), Galileo (EU), BeiDou (CN) and two regional systems – QZSS (Japan) and IRNSS or NavIC (India). Accuracy of the Position obtained from GNSS Data depends on the GNSS system used.

11.23.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/GNSSObject.json

11.23.4 Semantics

Label	Size	Description

Header	N1 Bytes	GNSS Object Header
– Standard-GNSSObject	9 Bytes	The characters “CAV-GNO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
GNSSObjectID	N5 Bytes	Identifier of the GNSS Object.
GNSSObjectSpaceTime	N6 Bytes	Space and Time info of Data Object.
GNSSObjectDataQualifier	N7 Bytes	GNSS Object Data Qualifier.
DescrMetadata	N8 Bytes	Descriptive Metadata

11.24 LiDAR Object

11.24.1 Definition

A Data Type including a collection of Basic LiDAR Objects.

A LiDAR Object can have a hierarchical structure where LiDAR Objects contain Basic LiDAR Objects and LiDAR Objects.

11.24.2 Functional Requirements

A LiDAR Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the LiDAR Object.
3. Space-Time information of the LiDAR Object.
4. Basic LiDAR Object and LiDAR Objects included in the LiDAR Objects.
5. Annotation data set including:
    1. Annotations
    2. Space-Times of the Annotations.
    3. Rights to perform Actions on the LiDAR Object.
6. The Rights that may be exercised on the LiDAR Object.

Note that.

1. A LiDAR Object that does not include Sub-Scenes and only one Basic LiDAR Object is a Basic LiDAR Object.
2. The Space-Time information of a Basic LiDAR Object and LiDAR Object included in a LiDAR Object may be superseded by the Space-Time information of the LiDAR Object containing them.

11.24.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/LiDARObject.json

11.24.4 Semantics

Label	Size	Description

Header	N1 Bytes	LiDAR Object Header
– Standard-LiDARObject	9 Bytes	The characters "OSD-LIO-V"
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character "."
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
LiDARObjectID	N5 Bytes	Identifier of the LiDAR Object.
LiDARObjectSpaceTime	N6 Bytes	Space-Time of LiDAR Object.
BasicLiDARObjectCount	N7 Bytes	Set of Parent LiDAR Objects.
BasicLiDARObjects[]	N8 Bytes	Set of Basic LiDAR Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic LiDAR Object in the LiDAR Object.
- BasicLiDARObject	N10 Bytes	A Basic LiDAR Object in the LiDAR Object.
LiDARObjectCount	N11 Bytes	Number of LiDAR Objects.
LiDARObjects[]	N12 Bytes	Set of LiDAR Objects.
- SpaceTime	N13 Bytes	Space Time of a LiDAR Object in the LiDAR Object.
- 3DModelObject	N14 Bytes	A LiDAR Object in the LiDAR Object
Annotations[]	N15 Bytes	Set of LiDAR Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.24.5Conformance Testing

A Data instance Conforms with LiDAR Object (OSD-LIO) V1.3 if:

1. The Data validates against the LiDAR Object's JSON Schema.
2. All Data in the LiDAR Object's JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.25 LiDAR Scene Descriptors

11.25.1Definition

A Data Type including the LiDAR Objects of a scene, their sub-scenes, and their arrangement in the scene.

11.25.2Functional Requirements

LiDAR Scene Descriptors include

1. LiDAR Objects
2. The Descriptors of the LiDAR Scenes includes in the LiDAR Scene called LiDAR Sub-Scenes.
3. Rights that may be exercised on the LiDAR Scene.

Scenes may be hierarchical, i.e., they may contain Objects and Scenes.

11.25.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/LiDARSceneDescriptors.json

11.25.4Semantics

Label	Size	Description
Header	N1 Bytes	LiDAR Scene Descriptors Header
- Standard-LiDARSceneDescriptors	9 Bytes	The characters "OSD-LSD-V"
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
SceneDescriptorsID	N5 Bytes	Identifier of Scene Descriptors.
SceneDescriptorsSpaceTime	N6 Bytes	Space and Time of Scene Descriptors.
ObjectCount	N7 Bytes	Number of Objects in Scene.
Objects[]	N8 Bytes	Set of Objects.
- Object or ObjectID	N9 Bytes	Object in the Scene of its ID.
- ObjectSpaceTime	N10 Bytes	Space Time of Object.
SubSceneCount	N11 Bytes	Number of Sub-Scenes in Scene.
SubScenes[]	N12 Bytes	Set of Sub-Scenes in the Scene.
- SubScene or SubSceneID	N13 Bytes	Sub-Scene in the Scene or its ID.
- SubSceneSpaceTime	N14 Bytes	Space Time of Sub-Scene.
DescrMetadata	N15 Bytes	Descriptive Metadata

11.25.5Conformance Testing

A Data instance Conforms with LiDAR Scene Descriptors (OSD-LSD) V1.3 if:

1. The Data validates against the Scene Descriptors' JSON Schema.
2. All Data in the Scene Descriptors' JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.26 Location

11.26.1Definition

A region of an entity with Space-Time attributes that is further subdivided in Basic Locations.

11.26.2Functional Requirements

A Location

1. Has an extension limited in Space and Time.
2. Is composed of Basic Locations, e.g.:
   1. A room can be a Basic Location of the Location defined as an apartment.
   2. An apartment can be a Basic Location of the Location defined as a building.

11.26.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/Location.json

11.26.4Semantics

Label	Size	Description
Header	N1 Bytes	Location Header
– Standard	9 Bytes	The characters "MMM-LOC-V"
– Version	N2 Bytes	Major version
– Dot-separator	1 Byte	The character "."
– Subversion	N3 Bytes	Minor version
MInstanceID	N4 Bytes	Identifier of M-Instance.
LocationID	N5 Bytes	Identifier of Location.
LocationData	N6 Bytes	Locations and Basic-Locations composing the Location.
– BasicLocation	N7 Bytes	A Basic Location composing the Location.
– Time	N8 Bytes	Time of validity of Basic Location in Location
DescrMetadata	N9 Bytes	Descriptive Metadata.

11.26.5Conformance Testing

A Data instance Conforms with Location (OSD-LOC) V1.2 if:

1. The Data validates against the Location's JSON Schema.
2. All Data in the Location's JSON Schema
   1. Have the specified type
   2. Validate against their JSON Schemas
   3. Conform with their Data Qualifiers if present.

11.27 Offline Map Object

11.27.1Definition

A Data Type including a collection of Basic Offline Map Objects.

An Offline Map Object can have a hierarchical structure where Offline Map Objects contain Basic Offline Map Objects and Offline Map Objects.

11.27.2Functional Requirements

An Offline Map Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the Offline Map Object.
3. Space-Time information of the Offline Map Object.
4. Basic Offline Map Object and Offline Map Objects included in the Offline Map Objects.
5. Annotation data set including:
    1. Annotations
    2. Space-Times of the Annotations.
    3. Rights to perform Actions on the Offline Map Object.
6. The Rights that may be exercised on the Offline Map Object.

Note that.

1. An Offline Map Object that does not include Sub-Scenes and only one Basic Offline Map Object is a Basic Offline Map Object.
2. The Space-Time information of a Basic Offline Map Object and Offline Map Object included in an Offline Map Object may be superseded by the Space-Time information of the Offline Map Object containing them.

11.27.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/OfflineMapObject.json.json

11.27.4*Semantics*

Label	**Size**	**Description**
Header	N1 Bytes	Offline Map Object Header
– Standard-AudioObject	9 Bytes	The characters “OSD-OMO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
OfflineMapObjectID	N5 Bytes	Identifier of the Offline Map Object.
OfflineMapObjectSpaceTime	N6 Bytes	Space-Time of Offline Map Object.
BasicOfflineMapObjectCount	N7 Bytes	Set of Parent Offline Map Objects.
BasicOfflineMapObjects[]	N8 Bytes	Set of Basic Offline Map Objects.

- SpaceTime	N9 Bytes	Space Time of a Basic Offline Map Object in the Offline Map Object.
- Basic OfflineMapObject	N10 Bytes	A Basic Offline Map Object in the Offline Map Object.
OfflineMapObjectCount	N11 Bytes	Number of Offline Map Objects.
OfflineMapObjects[]	N12 Bytes	Set of Offline Map Objects.
- SpaceTime	N13 Bytes	Space Time of an Offline Map Object in the Offline Map Object.
- AudioObject	N14 Bytes	An Offline Map Object in the Offline Map Object
Annotations[]	N15 Bytes	Set of Offline Map Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.27.5 Conformance Testing

A Data instance Conforms with Offline Map Object (OSD-AUO) V1.3 if:

1. The Data validates against the Offline Map Object's JSON Schema.
2. All Data in the Offline Map Object's JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.28 Offline Map Scene Descriptors

11.28.1 Definition

A Data Type including the Offline Map Objects of a scene, their sub-scenes, and their arrangement in the scene.

11.28.2 Functional Requirements

Offline Map Scene Descriptors include

1. Offline Map Objects
2. The Descriptors of the Offline Map Scenes includes in the Offline Map Scene called Offline Map Sub-Scenes.
3. Rights that may be exercised on the Offline Map Scene.

Scenes may be hierarchical, i.e., they may contain Offline Map Objects and Offline Map Scenes.

11.28.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/OfflineMapSceneDescriptors.json

11.28.4Semantics

Label	Size	Description
Header	N1 Bytes	Offline Map Scene Descriptors Header
- Standard-Offline MapSceneDescriptors	9 Bytes	The characters “OSD-OSD-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
SceneDescriptorsID	N5 Bytes	Identifier of Scene Descriptors.
SceneDescriptorsSpaceTime	N6 Bytes	Space and Time of Scene Descriptors.
ObjectCount	N7 Bytes	Number of Objects in Scene.
Objects[]	N8 Bytes	Set of Objects.
- Object or ObjectID	N9 Bytes	Object in the Scene of its ID.
- ObjectSpaceTime	N10 Bytes	Space Time of Object.
SubSceneCount	N11 Bytes	Number of Sub-Scenes in Scene.
SubScenes[]	N12 Bytes	Set of Sub-Scenes in the Scene.
- SubScene or SubSceneID	N13 Bytes	Sub-Scene in the Scene or its ID.
- SubSceneSpaceTime	N14 Bytes	Space Time of Sub-Scene.
DescrMetadata	N15 Bytes	Descriptive Metadata

11.28.5Conformance Testing

A Data instance Conforms with Offline Map Scene Descriptors (OSD-OSD) V1.3 if:

1. The Data validates against the Scene Descriptors’ JSON Schema.
2. All Data in the Scene Descriptors’ JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.29 Path

11.29.1Definition

Path is a sequence of Points of View.

11.29.2 Functional Requirements

Path is defined against an M-Instance or an OfflineMap.

11.29.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/Path.json

11.29.4 Semantics

Label	Size	Description
Header	N1 Bytes	Path Header
- Standard	9 Bytes	The characters "OSD-PAT-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
PathID	N4 Bytes	Identifier of Path.
MInstanceID	N5 Bytes	ID of the Virtual Space where Path is defined.
OfflineMapID	N6 Bytes	ID of the referenced OfflineMap.
PathData[]	N7 Bytes	Path Dataset.
- PointOfView	N8 Bytes	Planned Individual Point of View in the Path.
DescrMetadata	N9 Bytes	Descriptive Metadata

11.29.5 Conformance Testing

A Data instance Conforms with Path MPAI-OSD V1.3 (OSD-PAT) if the Data

1. Have the specified type
2. Validate against the Path's JSON Schema.

11.30 Point of View

11.30.1 Definition

Position and Orientation of an Object in a Virtual Environment excluding velocity and acceleration.

11.30.2 Functional Requirements

- An Object may have one of the following attributes: Speech, Audio; Visual; 3D Model, Audio-Visual; Haptic; Smell; RADAR; LiDAR; Ultrasound.
- Accuracy is the estimated absolute difference between the measured spatial and angular values of each of CartPosition, SpherPosition, Orientation, and their true value.

11.30.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/PointOfView.json

11.30.4 Semantics

Table 1 provides the semantics of the components of Point of View. The following should be noted:

1. Each of Position, Velocity, and Acceleration is provided either in Cartesian (X,Y,Z) or Spherical (r,φ,θ) Coordinates.
2. The Euler angles are indicated by (α,β,γ).

Table 1 – Semantics of Point of View

Label	Size	Description
Header	N1 Bytes	Point of View Header
- Standard-Point of View	9 Bytes	The characters "OSD-OPV-V"
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstance	N4 Bytes	ID id Virtual space Orientation refers tu
PointOfViewID	N5 Bytes	Identifier of Object Point of View.
General	N6 Bytes	Set of general data.
- CoordType	N7 Bytes	One of Cartesian, Spherical, Geodesic, Toroidal.
- ObjectType	N8 Bytes	One of Digital Human, Generic.
- MediaType	N9 Bytes	One of Speech, Audio, Visual, Audio-Visual, Haptic, Smell, RADAR, LiDAR, Ultrasound.
PositionAndOrientation		
- CartPosition (X,Y,Z)	N10 Bytes	Array (in metres)
- CartPositionAccuracy (X,Y,Z)	N11 Bytes	Array Of CartPositionAccuracy
- SpherPosition (r,φ,θ)	N12 Bytes	Array (in metres and degrees)
- SpherPositionAccuracy (r,φ,θ)	N13 Bytes	Array of - SpherPositionAccuracy
- Orient (α,β,γ)	N14 Bytes	Array (in degrees)
- OrientAccuracy (α,β,γ)	N15 Bytes	Array of OrientAccuracy
DescrMetadata	N16 Bytes	Descriptive Metadata

11.30.5 Conformance Testing

A Data instance Conforms with MPAI-OSD Point of View (OSD-OPV) V1.3 if:

1. The Data validates against the Point of View's JSON Schema.
2. All Data in the Point of View's JSON Schema.
    1. Have the specified type.
    2. Validate against their JSON Schemas.

11.31 Position

11.31.1Definition

A Data Type representing an Object's position, velocity, and acceleration.

11.31.2Functional Requirements

- The Position of an Object is that of a representative point in the Object.
- Cartesian and Polar Coordinate Systems are supported.
- An Object may have one of the following attributes: Speech, Audio; Visual; 3D Model, Audio-Visual; Haptic; Smell; RADAR; LiDAR; Ultrasound.
- Accuracy is the estimated absolute difference between the measured spatial values of each of CartPosition, SpherPosition, CartVelocity, SpherVelocity, CartAccel, SpherAccel and their true value.

11.31.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/Position.json

11.31.4Semantics

Table 1 provides the semantics of Position. It should be noted that each of Position, Velocity, and Acceleration can be expressed either in Cartesian (X,Y,Z) or Spherical (r,φ,θ) Coordinates.

Table 1 – Semantics of the Spatial Attitude

Label	Size	Description
Header	N1 Bytes	Position Header
- Standard-Position	9 Bytes	The characters "OSD-OPS-V"
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	ID of Virtual Space Position refers to.
ObjectPositionID	N5 Bytes	Identifier of Object Position.
General	N6 Bytes	Set of general data
- CoordinateType	N7 Bytes	One of Cartesian, Spherical, Geodesic, Toroidal.
- ObjectType	N8 Bytes	One of Digital Human, Generic.
- MediaType	N9 Bytes	One of Speech, Audio, Visual, 3D Model, Audio-Visual, Haptic, Smell, RADAR, LiDAR, Ultrasound.
Position		
- CartPosition (X,Y,Z)	N10 Bytes	Array (in metres)
- CartPositionAccuracy (X,Y,Z)	N11 Bytes	Array of CartPositionAccuracy
- SpherPosition (r,φ,θ)	N12 Bytes	Array (in metres and degrees)

- SpherPositionAccuracy (r,φ,θ)	N13 Bytes	Array of SpherPositionAccuracys
Velocity of Position		
- CartVelocity (X,Y,Z)	N14 Bytes	Array (in metres)
- CartVelocityAccuracy (X,Y,Z)	N15 Bytes	Array of - CartVelocityAccuracys (X,Y,Z)
- SpherVelocity (r,φ,θ)	N16 Bytes	Array (in metres and degrees)
- SpherVelocityAccuracy (r,φ,θ)	N17 Bytes	Array of SpherVelocityAccuracys
Acceleration of Position		
- CartAccel (X,Y,Z)	N18 Bytes	Array (in metres)
- CartAccelAccuracy (X,Y,Z)	N19 Bytes	Array of CartAccelAccuracys
- SpherAccel (r,φ,θ)	N20 Bytes	Array (in metres and degrees)
- SpherAccel (r,φ,θ)	N21 Bytes	Array (in metres and degrees)
DescrMetadata	N22 Bytes	Descriptive Metadata

11.31.5 Conformance Testing

A Data instance Conforms with MPAI-OSD V1.3 Position (OSD-OPS) if:

1. The Data validates against the Position 's JSON Schema.
2. All Data in the Position 's JSON Schema have the specifies type.

11.32 RADAR Object

11.32.1 Definition

A Data Type including a collection of Basic RADAR Objects.

A RADAR Object can have a hierarchical structure where RADAR Objects contain Basic RADAR Objects and RADAR Objects.

11.32.2 Functional Requirements

A RADAR Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the RADAR Object.
3. Space-Time information of the RADAR Object.
4. Basic RADAR Object and RADAR Objects included in the RADAR Objects.
5. Annotation data set including:
    1. Annotations
    2. Space-Times of the Annotations.
    3. Rights to perform Actions on the RADAR Object.
6. The Rights that may be exercised on the RADAR Object.

Note that.

1. A RADAR Object that does not include Sub-Scenes and only one Basic RADAR Object is a Basic RADAR Object.
2. The Space-Time information of a Basic RADAR Object and RADAR Object included in a RADAR Object may be superseded by the Space-Time information of the RADAR Object containing them.

11.32.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/RADARObject.json

11.32.4 Semantics

Label	Size	Description
Header	N1 Bytes	RADAR Object Header
– Standard-RADARObject	9 Bytes	The characters “OSD-AUO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
RADARObjectID	N5 Bytes	Identifier of the RADAR Object.
RADARObjectSpaceTime	N6 Bytes	Space-Time of RADAR Object.
BasicRADARObjectCount	N7 Bytes	Set of Parent RADAR Objects.
BasicRADARObjects[]	N8 Bytes	Set of Basic RADAR Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic RADAR Object in the RADAR Object.
- BasicRADARObject	N10 Bytes	A Basic RADAR Object in the RADAR Object.
RADARObjectCount	N11 Bytes	Number of RADAR Objects.
RADARObjects[]	N12 Bytes	Set of RADAR Objects.
- SpaceTime	N13 Bytes	Space Time of a RADAR Object in the RADAR Object.
- RADARObject	N14 Bytes	A RADAR Object in the RADAR Object
Annotations[]	N15 Bytes	Set of RADAR Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.32.5 Conformance Testing

A Data instance Conforms with RADAR Object (OSD-RAO) V1.3 if:

1. The Data validates against the RADAR Object’s JSON Schema.
2. All Data in the RADAR Object’s JSON Schema
    1. Have the specified type

2. Validate against their JSON Schemas
3. Conform with their Data Qualifiers.

11.33 RADAR Scene Descriptors

11.33.1 Definition

A Data Type including the RADAR Objects of a scene, their sub-scenes, and their arrangement in the scene.

In the following, Data, Qualifier, and Object should be read as RADAR Data, RADAR Qualifiers, and RADAR Object, respectively.

11.33.2 Functional Requirements

A Basic Object may include:

1. The ID of a Virtual Space (M-Instance) where it is or is intended to be located.
2. The ID of the Basic Object.
3. The ID(s) of Parent Object(s) supporting two cases:
    1. The Parent Object has spawned two (or more) Objects.
    2. Two (or more) Parent Objects have merged into one.
4. The Space-Time information of Parent Objects in an M-Instance.
5. The ID(s) of Child Object(s).
6. The Space-Time information of Child Objects in an M-Instance.
7. The Space-Time information of the Basic Object in an M-Instance.
8. The Qualifier of the specific Data Type.
9. The Rights that can be exercised on the Basic Object.
10. The set of Annotations including, for each Annotation:
    1. Space-Time information of the Annotation.
    2. Rights to perform Actions on the Annotation.

11.33.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/BasicRADARObject.json

11.33.4 Semantics

Label	Size	Description
Header	N1 Bytes	Basic RADAR Object Header
– Standard-BasicRADARObject	9 Bytes	The characters “OSD-BRO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
BasicObjectID	N5 Bytes	Identifier of the Basic Object.

BasicObjectSpaceTime	N6 Bytes	Space-Time info of the Basic Object.
Qualifier	N7 Bytes	Qualifier of Basic Data.
BasicObjectAnnotations[]	N8 Bytes	Annotations of Basic Object.
– Annotation	N9 Bytes	ID of Annotation
– AnnotationSpaceTime	N10 Bytes	Where/when Annotation is attached.
Rights	N11 Bytes	Rights to perform Actions of the Basic Object.
DescrMetadata	N12 Bytes	Descriptive Metadata

11.33.5Conformance Testing

A Data instance Conforms with Basic Object V1.3 if:

1. The Data validates against the Basic Object's JSON Schema.
2. All Data in the Basic Object's JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers, if present.

11.34 Road State

11.34.1 Definition

Road State is a collection of elements describing the state of the Road relevant to the Path traversed by a CAV. For each element of the Road State, the CAV records the value provided by:

1. Ego CAV, e.g., from the Motion Actuation Subsystem.
2. CAVs in range or other identified external sources.

11.34.2 Functional Requirements

Road State includes a subset of the following data:

1. Time the Road State was generated.
2. Validity Period
3. ID of CAV producing the Road State.
4. Segment to which Road State applies.
5. Road Attributes.
6. Weather Data
7. Submersion
8. Destruction
9. Pothole Position
10. Roadwork Position
11. Traffic Flow
12. Traffic Jam

11.34.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/RoadState.json

11.34.4 Semantics

Label	Size	Description
Header	N1 Bytes	Road State Header
- Standard - Road State	8 Bytes	The characters “CAV-RDS-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
RoadStateID	N4 Bytes	Identifier of the Road State.
RoadState	N5 Bytes	Set of Road State Data from multiple sources.
- RoadStateTime	N6 Bytes	Time of Road State creation and Validity Period.
- CAVID	N7 Bytes	ID of the CAV or CAV-Aware equipment providing the data.
- Segment	N8 Bytes	The Road Segment targeted by Road State
- Road Attributes	N9 Bytes	The Attributes of the Road Segment
- WeatherData	N10 Bytes	Weather Data from a source identified by ID and Pose.
- Submersion	M11 Bytes	Includes cm of water above road surface and Location.
- Destruction	N12 Bytes	Identified by Location
- Pothole	N13 Bytes	Identified by Location
- Works	N14 Bytes	Identified by Location
- TrafficFlow	N15 Bytes	Pose and traffic flow in the same and opposite direction (both in Vehicles/second).
-TrafficJam	N16 Bytes	Location
DescrMetadata	N17 Bytes	Descriptive Metadata

11.35 pace-Time

11.35.1Definition

Data Type representing the Spatial Attitude and Time information.

11.35.2Functional Requirements

Space-Time includes Spatial Attitude and Time.

11.35.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/SpaceTime.json

11.35.4 Semantics

Label	Size	Description
Header	N1 Bytes	Space-Time Header
- Standard-Object	9 Bytes	The characters "OSD-SPT-V"
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstance	N4 Bytes	Identifier of Virtual Space.
SpaceTimeID	N5 Bytes	Identifier of Space-Time.
Space	N6 Bytes	Spatial Attitudes at T_0 and T_1
Time	N7 Bytes	Time interval between T_0 and T_1
DescrMetadata	N8 Bytes	Descriptive Metadata

11.35.5 Conformance Testing

A Data instance Conforms with Space-Time (OSD-SPT) V1.3 if:

1. The Data validates against the Space-Time's JSON Schema.
2. All Data in the Space-Time's JSON Schema
    1. Have the specified type.
    2. Validate against their JSON Schemas.
    3. Conform with their Data Qualifiers if present.

11.36 Spatial Attitude

11.36.1 Definition

An Item representing the Position and Orientation of an Object, and their velocities and accelerations.

11.36.2 Functional Requirements

The Spatial Attitude is defined as the combination of Position and orientation, the Functional Requirements are defined by Position and Orientation.

11.36.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/SpatialAttitude.json

11.36.4 Semantics

Table 1 provides the semantics of the components of the Spatial Attitude.

Table 1 – Semantics of the Spatial Attitude

Label	Size	Description
Header	N1 Bytes	Spatial Attitude Header
- Standard-SpatialAttitude	9 Bytes	The characters “OSD-OSA-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	ID of Virtual Space Objectrefers to.
ObjectSpatialAttitudeID	N5 Bytes	Identifier of Object Spatial Attitude.
General	N6 Bytes	Set of general data
- CoordinateType	N7 Bytes	One of Cartesian, Spherical, Geodesic, Toroidal.
- ObjectType	N8 Bytes	One of Digital Human, Generic.
- MediaType	N9 Bytes	One of Speech, Audio, Visual, Audio-Visual, Haptic, Smell, RADAR, LiDAR, Ultrasound.
Position	N10 Bytes	As specified by Position
Orientation	N11 Bytes	As specified by Orientation
DescrMetadata	N20 Bytes	Descriptive Metadata

11.36.5 Conformance Testing

A Data instance Conforms with V1.2 Spatial Attitude V1.3 (OSD-OSA) if:

1. The Data validates against the Spatial Attitude’s JSON Schema.
2. All Data in the Spatial Attitude ’s JSON Schema have the specified type.

11.37 Time

11.37.1 Definition

The start time and the end time of a duration.

11.37.2 Functional Requirements

Origin of Time can be Absolute (from 1970/01/01) or relative to a user-selected value.

11.37.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/Time.json

11.37.4 Semantics

Label	Size	Description
Header	N1 Bytes	Time Header
- Standard-Object	9 Bytes	The characters “OSD-TIM-V”
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance
TimeID	N5 Bytes	Identifier of M-Instance.
TimeData	17 Bytes	Data about Time
- TimeType	0 bit	0=Relative: start at 0000/00/00T00:00 1=Absolute: start at 1970/01/01T00:00.
- TimeUnit	1-5	reserved
- Reserved	6-7 bits	00=seconds, 01=milliseconds, 10=microseconds, 11=nanoseconds.
- StartTime	8 Bytes	Start of Time.
- EndTime	8 Bytes	End of Time.
DescrMetadata	N6 Bytes	Descriptive Metadata

11.37.5 Conformance Testing

A Data instance Conforms with MPAI-OSD Time V1.3 (OSD-) if:

1. The Data validates against the Times’s JSON Schema.
2. All Data in the Times’s JSON Schema have the specified type.

11.38 Traffic Rules

11.38.1 Definition

A representation of the semantics of the Traffic Rules as identified by the traffic signs of the Vienna Convention.

11.38.2 Functional Requirements

The Traffic Rules Data Type only provides the semantics of traffic signs, not their visual representation which is assumed to be included in the device performing sign recognition.

The Vienna Convention on Road Traffic is used.

11.38.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/TrafficaRules.json

11.38.4 Semantics

Label	Size	Description
Header	N1 Bytes	CAV State Header
- Standard	9 Bytes	The characters “CAV-TRR-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
Traffic Rules	N4 Bytes	Vienna Convention on Road Traffic
- Road Signs	N5 Bytes	Vienna Convention on Road Traffic
- Danger warnings	N6 Bytes	Vienna Convention on Road Traffic
- Dangerous bend(s)	N7 Bytes	Vienna Convention on Road Traffic
- Other dangers	N8 Bytes	Vienna Convention on Road Traffic
- Priority signs	N9 Bytes	Vienna Convention on Road Traffic
- Prohibitory or restrictive signs	N10 Bytes	Vienna Convention on Road Traffic
- Prohibition and restriction of entry	N11 Bytes	Vienna Convention on Road Traffic
- Prohibited for a certain category of vehicle or road-user	N12 Bytes	Vienna Convention on Road Traffic
- Prohibition of turning	N13 Bytes	Vienna Convention on Road Traffic
- Prohibition of U-turns	N14 Bytes	Vienna Convention on Road Traffic
- Prohibition of overtaking	N15 Bytes	Vienna Convention on Road Traffic
- Speed limit	N16 Bytes	Vienna Convention on Road Traffic
- Prohibition of the use of audible warning devices	N17 Bytes	Vienna Convention on Road Traffic
- Prohibition of passing without stopping	N18 Bytes	Vienna Convention on Road Traffic
- End of prohibition or restriction	N19 Bytes	Vienna Convention on Road Traffic
- Prohibition or restriction of standing and parking	N20 Bytes	Vienna Convention on Road Traffic
- Mandatory signs	N21 Bytes	Vienna Convention on Road Traffic
- Direction to be followed	N22 Bytes	Vienna Convention on Road Traffic
- Pass this side	N23 Bytes	Vienna Convention on Road Traffic
- Compulsory roundabout	N24 Bytes	Vienna Convention on Road Traffic
- Compulsory cycle track	N25 Bytes	Vienna Convention on Road Traffic
- Compulsory footpath	N26 Bytes	Vienna Convention on Road Traffic
- Compulsory track for riders on horseback	N27 Bytes	Vienna Convention on Road Traffic
- Compulsory minimum speed	N28 Bytes	Vienna Convention on Road Traffic
- End of compulsory minimum speed	N29 Bytes	Vienna Convention on Road Traffic
- Snow chains compulsory	N30 Bytes	Vienna Convention on Road Traffic
- Compulsory direction for vehicles carrying dangerous goods	N31 Bytes	Vienna Convention on Road Traffic
- Special regulation signs	N32 Bytes	Vienna Convention on Road Traffic

- Signs indicating a regulation or danger warning applying to one or more traffic lanes:	N33 Bytes	Vienna Convention on Road Traffic
- Signs indicating lanes reserved for buses	N34 Bytes	Vienna Convention on Road Traffic
- ONE-WAY sign	N35 Bytes	Vienna Convention on Road Traffic
- Preselection sign	N36 Bytes	Vienna Convention on Road Traffic
- Signs notifying an entry to or an exit from a motorway	N37 Bytes	Vienna Convention on Road Traffic
- Signs notifying an entry to or exit from a road on which the traffic rules are the same as on a motorway	N38 Bytes	Vienna Convention on Road Traffic
- Signs indicating the beginning and the end of a built-up area	N39 Bytes	Vienna Convention on Road Traffic
- Signs having zonal validity	N40 Bytes	Vienna Convention on Road Traffic
- Signs notifying the entry to or exit from a tunnel where special rules apply	N41 Bytes	Vienna Convention on Road Traffic
- PEDESTRIAN CROSSING sign	N42 Bytes	Vienna Convention on Road Traffic
- HOSPITAL sign	N43 Bytes	Vienna Convention on Road Traffic
- PARKING sign	N44 Bytes	Vienna Convention on Road Traffic
- Signs notifying a bus or tramway stop	N45 Bytes	Vienna Convention on Road Traffic
- Signs indicating a stopping place in case of emergency or danger	N46 Bytes	Vienna Convention on Road Traffic
- Information, facilities or service signs	N47 Bytes	Vienna Convention on Road Traffic
- FIRST-AID STATION symbol	N48 Bytes	Vienna Convention on Road Traffic
- Miscellaneous symbols	N49 Bytes	Vienna Convention on Road Traffic
- Direction, position or indication signs	N50 Bytes	Vienna Convention on Road Traffic
- Advance direction signs	N51 Bytes	Vienna Convention on Road Traffic
- General case	N52 Bytes	Vienna Convention on Road Traffic
- Special cases	N53 Bytes	Vienna Convention on Road Traffic
- Direction signs	N54 Bytes	Vienna Convention on Road Traffic
- Confirmatory signs	N55 Bytes	Vienna Convention on Road Traffic
- Indication signs	N56 Bytes	Vienna Convention on Road Traffic
- Signs indicating the number and direction of traffic lanes	N57 Bytes	Vienna Convention on Road Traffic
- Signs indicating closure of a traffic lane	N58 Bytes	Vienna Convention on Road Traffic
- Sign notifying advised itinerary for heavy vehicles	N59 Bytes	Vienna Convention on Road Traffic
- Sign notifying an escape lane	N60 Bytes	Vienna Convention on Road Traffic
- Other signs	N61 Bytes	Vienna Convention on Road Traffic
- Additional Panels	N62 Bytes	Vienna Convention on Road Traffic
- Priority signs	N63 Bytes	Vienna Convention on Road Traffic

- Traffic light	N64 Bytes	Vienna Convention on Road Traffic
- Signals for Vehicular traffic	N65 Bytes	Vienna Convention on Road Traffic
- Non flashing lights	N66 Bytes	Vienna Convention on Road Traffic
- Flashing lights	N67 Bytes	Vienna Convention on Road Traffic
- Road marking	N68 Bytes	Vienna Convention on Road Traffic
- Longitudinal marking	N69 Bytes	Vienna Convention on Road Traffic
- Transversal marking	N70 Bytes	Vienna Convention on Road Traffic
- Markings on the carriageway	N71 Bytes	Vienna Convention on Road Traffic
- Danger warning	N72 Bytes	Vienna Convention on Road Traffic
- Dangerous bend(s)	N73 Bytes	Vienna Convention on Road Traffic
- Level crossing	N74 Bytes	Vienna Convention on Road Traffic
DescrMetadata	N75 Bytes	Descriptive Metadata

11.39 Traffic Sign Objects

11.39.1 Definition

Representation of Traffic Sign-related objects of a Location.

11.39.2 Functional Requirements

The types of Traffic Sign-related objects of a Location are:

1. Traffic Policeman
2. Horizontal Road Signs on the road (lanes, turn right/left, one way, stop signs, words).
3. Vertical Road Signs above the road (signs on poles, signs on objects).
4. Traffic lights
5. Audio Signs (siren, whistle, horn, uttered words).

11.39.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/TrafficSignObjects.json

11.39.4 Semantics

Label	Size	Description
Header	N1 Bytes	Traffic Sign Objects Header
- Standard-TrafficSignObjects	9 Bytes	The characters "CAV-TSO-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
TrafficSignObjectsID	N4 Bytes	Identifier of Traffic Sign Objects.
TrafficSignObjects[]	N5 Bytes	Traffic Sign Object dataset.
- Location	N6 Bytes	Location containing Traffic Sign Objects.
- TrafficObjects[]	N7 Bytes	Datasets of Traffic Objects.

- TrafficObject	N9 Bytes	A specific Traffic Object.
- SpatialAttitude	N10 Bytes	Traffic Object's Spatial Attitude.
- TextObjects[]	N11 Bytes	Text Objects related to the Traffic Object.
- TextObject	N12 Bytes	A specific Text Object.
- SpatialAttitude	N13 Bytes	Text Object's Spatial Attitude.
- PriorityInformation	N14 Bytes	One of Police, Ambulance, Hazard
DescrMetadata	N15 Bytes	Descriptive Metadata.

11.40 Trajectory

11.40.1Definition

The sequence of start and end Spatial Attitudes *SA* (SA_1, SA_2, …, SA_i) and corresponding Times *t* (t_1, t_2, t_j) expected and actual of a series of segments.

11.40.2Functional Requirements

A Trajectory is composed of Segments. Each Segment is described by the expected and actual start and end Spatial Attitudes and Times.

11.40.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/Trajectory.json

11.40.4Semantics

Label	Size	Description
Header	N1 Bytes	Trajectory Header
- Standard-Trajectory	9 Bytes	The characters "CAV-TRJ-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
TrajectoryID	N4 Bytes	Identifier of Trajectory.
TrajectoryData[]	N5Bytes	Data in the Trajectory
- SpaceTime	N6 Bytes	Expected and/or actual Spatial Attitude and Time of a Trajectory segment.
DescrMetadata	N7 Bytes	Descriptive Metadata

11.40.5Conformance Testing

A Data instance Conforms with Trajectory (OSD-TRJ) V1.3 if:

1. The Data validates against the Trajectory 's JSON Schema.
2. All Data in the Trajectory 's JSON Schema

   1. Have the specified type
   2. Validate against JSON Schemas.
   3. Conform with their Data Qualifiers if present.

11.41 Ultrasound Object

11.41.1Definition

A Data Type including a collection of Basic Ultrasound Objects.

Ann Ultrasound Object can have a hierarchical structure where Ultrasound Objects contain Basic Ultrasound Objects and Ultrasound Objects.

11.41.2Functional Requirements

A Ultrasound Object may include:

1. ID of a Virtual Space (M-Instance) where it is or intended to be located.
2. ID of the Ultrasound Object.
3. Space-Time information of the Ultrasound Object.
4. Basic Ultrasound Object and Ultrasound Objects included in the Ultrasound Objects.
5. Annotation data set including:
   1. Annotations
   2. Space-Times of the Annotations.
   3. Rights to perform Actions on the Ultrasound Object.
6. The Rights that may be exercised on the Ultrasound Object.

Note that.

1. An Ultrasound Object that does not include Sub-Scenes and only one Basic Ultrasound Object is a Basic Ultrasound Object.
2. The Space-Time information of a Basic Ultrasound Object and Ultrasound Object included in a Ultrasound Object may be superseded by the Space-Time information of the Ultrasound Object containing them.

11.41.3Syntax

https://schemas.mpai.community/OSD/V1.3/data/UltrasoundObject.json

11.41.4Semantics

Label	Size	Description
Header	N1 Bytes	Ultrasound Object Header
– Standard-UltrasoundObject	9 Bytes	The characters “OSD-USO-V”
– Version	N2 Bytes	Major version – 1 or 2 characters
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
UltrasoundObjectID	N5 Bytes	Identifier of the Ultrasound Object.

UltrasoundObjectSpaceTime	N6 Bytes	Space-Time of Ultrasound Object.
BasicUltrasoundObjectCount	N7 Bytes	Set of Parent Ultrasound Objects.
BasicUltrasoundObjects[]	N8 Bytes	Set of Basic Ultrasound Objects.
- SpaceTime	N9 Bytes	Space Time of a Basic Ultrasound Object in the Ultrasound Object.
- BasicUltrasoundObject	N10 Bytes	A Basic Ultrasound Object in the Ultrasound Object.
UltrasoundObjectCount	N11 Bytes	Number of Ultrasound Objects.
UltrasoundObjects[]	N12 Bytes	Set of Ultrasound Objects.
- SpaceTime	N13 Bytes	Space Time of a Ultrasound Object in the Ultrasound Object.
- UltrasoundObject	N14 Bytes	An Ultrasound Object in the Ultrasound Object
Annotations[]	N15 Bytes	Set of Ultrasound Object Annotation.
– Annotation	N16 Bytes	An Annotation.
– AnnotationSpaceTime	N17 Bytes	Where Annotation is attached and when it will be active.
– Rights	N18 Bytes	Actions that may be performed on the Annotation
Rights	N19 Bytes	Actions that may be performed on the Object.
DescrMetadata	N20 Bytes	Descriptive Metadata

11.41.5Conformance Testing

A Data instance Conforms with Ultrasound Object (OSD-USO) V1.3 if:

1. The Data validates against the Ultrasound Object’s JSON Schema.
2. All Data in the Ultrasound Object’s JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.42 Ultrasound Scene Descriptors

11.42.1Definition

A Data Type including the Ultrasound Objects of a scene, their sub-scenes, and their arrangement in the scene.

11.42.2Functional Requirements

Ultrasound Scene Descriptors include

1. Ultrasound Scene Objects
2. The Descriptors of the Ultrasound Scenes includes in the Ultrasound Scene called Ultrasound Sub-Scenes.
3. Rights that may be exercised on the Ultrasound Scene.

Scenes may be hierarchical, i.e., they may contain Ultrasound Objects and Ultrasound Scenes.

11.42.3 Syntax

https://schemas.mpai.community/OSD/V1.3/data/UltrasoundSceneDescriptors.json

11.42.4 Semantics

Label	Size	Description
Header	N1 Bytes	Ultrasound Scene Descriptors Header
- Standard-UltrasoundSceneDescriptors	9 Bytes	The characters "OSD-USD-V"
- Version	N2 Bytes	Major version – 1 or 2 characters
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 characters
MInstanceID	N4 Bytes	Identifier of M-Instance.
SceneDescriptorsID	N5 Bytes	Identifier of Scene Descriptors.
SceneDescriptorsSpaceTime	N6 Bytes	Space and Time of Scene Descriptors.
ObjectCount	N7 Bytes	Number of Objects in Scene.
Objects[]	N8 Bytes	Set of Objects.
- Object or ObjectID	N9 Bytes	Object in the Scene of its ID.
- ObjectSpaceTime	N10 Bytes	Space Time of Object.
SubSceneCount	N11 Bytes	Number of Sub-Scenes in Scene.
SubScenes[]	N12 Bytes	Set of Sub-Scenes in the Scene.
- SubScene or SubSceneID	N13 Bytes	Sub-Scene in the Scene or its ID.
- SubSceneSpaceTime	N14 Bytes	Space Time of Sub-Scene.
DescrMetadata	N15 Bytes	Descriptive Metadata

11.42.5 Conformance Testing

A Data instance Conforms with Ultrasound Scene Descriptors (OSD-USD) V1.3 if:

1. The Data validates against the Scene Descriptors' JSON Schema.
2. All Data in the Scene Descriptors' JSON Schema
    1. Have the specified type
    2. Validate against their JSON Schemas
    3. Conform with their Data Qualifiers.

11.43 AMS-MAS Message

11.43.1 Definition

A Message sent to:

1. The MAS by the AMS.

2. the AMS from the MAS.

11.43.2 Functional Requirements

The AMS issues an AMS Message to request that the MAS move the CAV along the Trajectory.This contains:

1. Trajectory
2. One of the two Actions:
    1. *Execute* the *Trajectory* to change the CAV's Spatial Attitude SA_A at time t_A to Spatial Attitude SA_B at time t_B.
    2. *Suspend* the *Execute Action.*

Upon receiving an AMS-MAS Message, the MAS implements the requests and issues a series of AMS-MAS Messages at intermediate Poses informing the AMS about the progress in the execution of the AMS-MAS Message.

Messages contain:

1. Current Time
2. CAV Spatial Attitude derived from MAS-internal sensors.
3. Road State in case of serious misalignment between the expected and the actual Spatial Attitude at the current Time, such as caused by Ice Conditions and Submersion.
4. CAV State.

11.43.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/AMSMASMessage.json

11.43.4 Semantics

Label	Size	Description
Header	N1 Bytes	AMS-MAS Message Header
- Standard - AMSMASMessage	8 Bytes	The characters "CAV-AMM-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
AMSMASMessageID	N4 Bytes	Identifier of AMS-MAS Message.
AMSMessage	N5 Bytes	Data in AMS-MAS Message.
- Time	N6 Bytes	Duration the Message applies to.
- Trajectory	N7 Bytes	Trajectory to be executed.
- Command	N8 Bytes	One of Execute, Suspend, Resume, Change.
MASMessage	N9 Bytes	Data in AMS-MAS Message.
- RoadState	N10 Bytes	Current Road State.

- CAVState	N11 Bytes	Current CAV State.
DescrMetadata	N12 Bytes	Descriptive Metadata

11.44 AMS Data

11.44.1 Definition

A Data Type representing the Data stored in the AMS Decision Recording AIM provided to an external device.

11.44.2 Functional Requirements

The AMS Recording Data includes:

1. Time
2. Route
3. Path
4. Trajectory
5. Road State
6. CAV State
7. Full Environment Descriptors.
8. AMS-MAS Messages.
9. Alert.

11.44.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/AMSData.json

11.44.4 Semantics

Label	Size	Description
Header	N1 Bytes	AMS Data Header
- Standard-AMSData	9 Bytes	The characters “CAV-AMD-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
AMSDataID	N4 Bytes	Identifier of AMS Recording Data instance.
AMSData	N5 Bytes	Set of Data in AMS Recording Data.
- Route	N6 Bytes	Route from CAV-RSP.
- Path	N7 Bytes	Path from CAV-PRP.
- Trajectory	N8 Bytes	Trajectory from CAV-MSP.
- Alert	N9 Bytes	Alert from ESS
- Road State	N10 Bytes	Road State from CAV-AMI

- CAV State	N11 Bytes	Road State from CAV-AMI
- Full Environment Descriptors	N12 Bytes	Full Environment Descriptors
- AMS-MAS Messages	N13 Bytes	AMS-MAS Messages to/from MAS
DescrMetadata	N14 Bytes	Descriptive Metadata

11.45 CAV Identifier

11.45.1 Definition

A code uniquely identifying a CAV instance. The CAV ID may be temporary.

11.45.2 Functional Requirements

The CAV identification system may carry the following information:

1. Country where the CAV was registered (optional).
2. Registration number in the country (optional).
3. CAV manufacturer identifier.
4. CAV model identifier.
5. M-Instance Identifier (optional).

The governance of CAV Identifiers is a vital element. However, it is out of scope of this Technical Specification.

11.45.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/CAVIdentifier.json

11.45.4 Semantics

Label	Size	Description
Header	N1 Bytes	
- Standard	9 Bytes	The characters “CAV-CID-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
CAVIdentifier	N4 Bytes	Identifier of CAV instance.
CAVIdentifierData	N5 Bytes	Set of Data in CAV Identifier
- CountryID	2 Bytes	2-character country identifier
- RegistrationID	N6 Bytes	CAV Registration ID in country
- ManufacturerID	N7 Bytes	Manufacturer ID
- ModelID	N8 Bytes	Model ID
- MInstanceID	N9 Bytes	Identifier of CAV’s M-Instance.
DescrMetadata	N10 Bytes	Descriptive Metadata

11.46 CAV State

11.46.1 Definition

A Description of the state of the CAV generated by the CAV's AMS using: information available inside the CAV as assessed by the CAV.

11.46.2 Functional Requirements

A CAV State includes the following information:

1. Time and Position of CAV State generation.
2. Battery state (temperature overload, insufficient capacity).
3. Brake responsiveness (measured by ineffective deceleration).
4. Motor Responsiveness (measured by ineffective acceleration).
5. Wheel responsiveness (measured by loss of traction, mechanical disfunction).

By using CAV State, the Autonomous Motion Subsystem (AMS) can estimate time and distance of operation assuming it has the values of Route, Lights, Air Conditioning, Cabin Temperature and Velocity.

11.46.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/CAVState.json

11.46.4 Semantics

Label	Size	Description
Header	N1 Bytes	CAV State Header
- Standard - CAVState	9 Bytes	The characters "CAV-CST-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
CAVStateID	N4 Bytes	Identifier of CAV State instance.
CAVState	N5 Bytes	Set of CAV State Data
- CAVStateSpaceTime	N6 Bytes	Spatial Attitude and Time of CAV State generation
- BatteryState	N7 Bytes	Measured in milliAmp-hours (mAh) or Watt-hours (Wh) and percentage of total battery charge.
- Brake Responsiveness	N8 Bytes	Fully developed deceleration computed according to reference
- Motor Responsiveness	N9 Bytes	Acceleration where **Acceleration (m/s2) = motor torque (Nm) × gear ratio / wheel radius (m) / mass (kg)**, and torque is measured in Nm, mass in kg, and radius in m.

- Wheel Responsiveness	N10 Bytes	Measured in degrees per unit angular momentum (kgm^2/s)
DescrMetadata	N13 Bytes	Descriptive Metadata

11.47 Ego-Remote AMS Message

11.47.1 Description

Message exchanged by the Ego CAV's and the Remote CAV's AMSs in the form of:

1. Request/Response of M-Location corresponding to the intended U-Location:
2. Transmission of the Trajectory the CAV intends to adopt.

11.47.2 Functional Requirements

The interaction between the Ego and Remote CAVs unfolds as follows:

1. A CAV sends an Ego-Remote AMS Message requesting information about how the Remote CAV understands a specific subset of its Environment corresponding to the intended U-Location.
2. The Remote CAV accepting the request:
    1. Converts the requested U-Location to the M-Location of its Full Environment Descriptors (FED).
    2. Extracts the subset of the FED corresponding to the M-Location thanks to the scene-based object description of the FED.
    3. Harvests available bandwidth to send a version of the FED that is compatible with the currently available mobile bandwidth
    4. Sends the requested M-Location with the level of detail defined in 3.
3. The Ego CAV
    1. Reconciles the different values of the components of its own M-Location and those received.
    2. Records in MAS Data major discrepancies between its own Positions and the one deduced from the Remote CAV FED.

CAV-TEC V1.0 does not consider the potential requirements to hide the identity of the CAV sending information extracted from its own Full Environment Descriptors and potential solutions to cope with that requirement.

11.47.3 Syntax

https://schems.mpai.community/CAV2/V1.0/data/EgoRemoteAMSMessage.json

11.47.4 Semantics

Label	Size	Description
Header	N1 Bytes	Road State Header
- Standard	9 Bytes	The characters “MMM-ERA-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes

- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
EgoRemoteAMSMessageID	N4 Bytes	Identifier of Ego-Remote-AMS Message.
RequestMessage	N5 Bytes	CAV's Request
- EgoCAVID	N6 Bytes	CAV ID
-ULocation	N7 Bytes	Specification of Remote Cav's Environment portion.
ResponseMessage	N8 Bytes	Data of Ego-Remote-AMS Message
- RemoteCAVID	N9 Bytes	M-Location in Remote AMS.
- RemoteFED	N10 Bytes	Subset of FED relevant to requested U-Location.
SenderInfoMessage	N11 Bytes	Message sent for information
- EgoCAVID	N12 Bytes	CAV ID
- Trajectory	N13 Bytes	Trajectory Ego AMS intends to adopt.
ReceiverInfoMessage	N14 Bytes	Message sent for information
- EgoCAVID	N15 Bytes	CAV ID
- Accept/Reject	N16 Bytes	Trajectory accepted/rejected by receiving Remote AMS
AMSAlertMessage	N17 Bytes	Messages alerting CAVs about Ego CAV status.
- EgoCAVID	N18 Bytes	CAV ID.
- AMSAlertMessage	N19 Bytes	One of Ambulance, Authority, Health, Evacuation Messages.
- Road State	N20 Bytes	Appropriate subset of Road State.
- CAV State	N21 Bytes	Appropriate subset of CAV State.
DescrMetadata	N22 Bytes	Descriptive Metadata

11.48 Full Environment Descriptors

11.48.1 Definition

The Full Environment Descriptors (FED) is the result of the Autonomous Motion Subsystem's *integration* of:

1. The BED from the Ego CAV's ESS.
2. The Road State and CAV State.
3. FED-related information received from Remote AMSs in range or Roadside Units.

11.48.2 Functional Requirements

The FED is generated from the BED with the following characteristics:

1. The Road State is added to the BED.

2. The CAV State is added to the BED as an Annotation to the Ego CAV.
3. Information from Remote CAVs or other CAV-enabled equipment is used to:
    1. Add to improve on or replace existing or missing information in the BED.
    2. Record in MAS Memory irreconcilable differences in the Ego-CAV Spatial Attitude between the Ego CAV's measurements and the values deduced from Remote CAVs.
4. Replaced or complemented objects retain the Device ID of the remote CAV's device that provided the information used by the Remote CAV to create the object.
5. The actual shape of the CAV may replace the existing representation, e.g., because the shape is derived from the CAV's Model ID.
6. The FED has a scalable representation allowing for:

    - Refinement of FED when new EST-specific Scene Descriptors are added.
    - Extraction of a FED subset based on a required Level of Detail in the form of, e.g., Object bounding boxes and their Spatial Attitudes.
    - Addition of new data, e.g., the shape of an Object improving on a previous 2D or bounding box information.
    - Fast access to Object metadata, such as Spatial Attitude and shape (e.g., bounding box for a Visual Object).
    - Update of Objects and Scenes from one Scene to another.

The AMS may:

1. Communicate the FED to the ESS.
2. Communicate a subset of Ego FED to other CAVs with different levels of detail, e.g., starting from Position and Bounding Box, depending on the available bandwidth.
3. Verify the feasibility of a Trajectory, e.g., to enable the AMS to check that the intended Trajectory of the Ego CAV does not collide with other Objects in the Decision Horizon or planned Trajectories communicated by Remote CAVs.

11.48.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/FullEnvironmentDescriptors.json

11.48.4 Semantics

Label	Size	Description
Header	N1 Bytes	Full Environment Descriptors Headers
- Standard-FullEnvironmentDescriptors	9 Bytes	The characters “CAV-FED-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
MInstanceID	N4 Bytes	ID of Virtual Space of the BED.
FullEnviroment DescriptorsID	N5 Bytes	FED ID.
AudioVisualSceneDescriptors	N6 Bytes	AV Scene Descriptors of Environment.

WeatherData	N7 Bytes	Weather Data integrated in Scene Description.
RoadState	N8 Bytes	Road State integrated in Scene Description.
CAVState	N9 Bytes	CAV State integrated in Scene Description.
DescrMetadata	N10 Bytes	Descriptive Metadata.

11.49 Road Attributes

11.49.1 Definition

A Data Type representing the features of a Road.

11.49.2 Functional Requirements

The features considered are:

- NumberOfLanes
- Length
- Width
- MaxSpeed
- MinSpeed
- MaxHeight
- MaxWeight
- LaneUsage (forward, backward)
- Category (oneway, toll, link)
- Types (highway, street, avenue, boulevard, lane)

11.49.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/RoadAttributes.json

11.49.4 Semantics

Label	Size	Description
Header	N1 Bytes	Road Attributes Header
– Standard-RoadAttributes	9 Bytes	The characters “CAV-RDA-V”
– Version	N2 Bytes	Major version – 1 or 2 Bytes
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
RoadAttributesID	N4 Bytes	Identifier of AMS Recording Data instance.
RoadAttributesTime	N5 Bytes	Time of RoadAttributes provisining.
RoadAttributes	N6 Bytes	Set of Data in AMS Recording Data.
– NumberOfLanes	N7 Bytes	Number of lanes
– Length	N8 Bytes	Length of Road
– Width	N9 Bytes	Width of Road
– MaxSpeed	N10 Bytes	Maximum vehicle speed allowed
– MinSpeed	N11 Bytes	Minimum vehicle speed allowed
– MaxHeight	N12 Bytes	Maximum vehicle height allowed

– MaxWeight	N13 Bytes	Maximum vehicle weight allowed
– LaneUsage	N14 Bytes	One of forward, backward
– Category	N15 Bytes	One of oneway, toll, link
– Types	N16 Bytes	One of highway, street, avenue, boulevard, lane
DescrMetadata	N17 Bytes	Descriptive Metadata

11.50 Route

11.50.1 Definition

A sequence of Way Points on the Offline Map with attached Start Times and Arrival Times. A Route may also be used to record information about the relevant Times and Places where the CAV made a stop.

11.50.2 Functional Requirements

Route is a series of Way Points on a specified Offline Map connected by roads. A Waypoint includes the local Road State.

11.50.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/Route.json

11.50.4 Semantics

Label	Size	Description
Header	N1 Bytes	Route Header
- Standard	9 Bytes	The characters “CAV-RTE-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
OfflineMapID	N4 Bytes	ID of the Offline map the Route refers to.
RouteID	N5 Bytes	Identifier of Route.
RouteData[]	N6 Bytes	Route Data set
- WayPointID	N7 Bytes	Identifier of a Way Point #n.
- EstimatedArrDepSpaceTime	N8 Bytes	Estimated Time when Way Point #n is is reached.
- ActualArrDepSpaceTime	N9 Bytes	Actual Time when Way Point #n is reached.
- SegmentState	N10 Bytes	Actual Road State information at Way Point #n.
DescrMetadata	N113 Bytes	Descriptive Metadata

11.51 Brake Command

11.51.1 Definition

The command issued by the Motion Actuation Subsystem to a Brake to reduce the speed of the CAV.

11.51.2 Functional Requirements

A Brake Command is expressed as the velocity a CAV should have after a Time since it application.

11.51.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/BrakeCommand.json

11.51.4 Semantics

Label	Size	Description
Header	N1 Bytes	Brake Command Header
- Standard	9 Bytes	The characters "CAV-BRC-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
BrakeCommandID	N4 Bytes	Format ID of BrakeCommand.
BrakeID	N6 Bytes	Identifier of Brake.
BrakeCommand	N7 Bytes	Set of BrakeCommands.
- TargetVelocity	N8 Bytes	Expected velocity at the end of duration.
- BrakeCommandTime	N9 Bytes	Duration of BrakeCommand application.
DescrMetadata	N10 Bytes	Descriptive Metadata

11.52 Brake Response

11.52.1 Definition

The response issued by a Brake to the Motion Actuation Subsystem informing about the result of the execution of a Brake Command.

11.52.2 Functional Requirements

The Response of a Brake is represented by:

1. Time the Brake Response is issued.
2. Velocity reached at that Time.
3. A particular Brake State

11.52.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/BrakeResponse.json

11.52.4 Semantics

Label	Size	Description
Header	N1 Bytes	Brake Response Header
- Standard-BrakeResponse	9 Bytes	The characters "CAV-BRR-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes

- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
BrakeID	N4 Bytes	Identifier of Brake.
BrakeResponse	N5 Bytes	Set of Brake Responses.
- BrakeResponseTime	N6 Bytes	Time of Brake Response.
- VelocityReached	N7 Bytes	Velocity reached after Brake action.
- BrakeState	N8 Bytes	One of the Brake State values.
DescrMetadata	N9 Bytes	Descriptive Metadata

11.53 Motor Command

11.53.1 Definition

The command issued by the Motion Actuation Subsystem to the Motor of a Wheel to enable the CAV to reach an assigned velocity after Time.

11.53.2 Functional Requirements

A Motor Command expresses

1. The target velocity.
2. The Time after which the target velocity should be reached.

11.53.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/MotorCommand.json

11.53.4 Semantics

Label	Size	Description
Header	N1 Bytes	Motor Command Header
- Standard - MotorCommand	9 Bytes	The characters "CAV-MRC-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
MotorCommandID	N4 Bytes	ID of Motor Command.
MotorID	N5 Bytes	Identifier of Motor.
MotorCommand	N7 Bytes	Set of Motor Command.
- Duration	N8 Bytes	Time during which torque is applied.
- TargetVelocity	N9 Bytes	Velocity that should be reached after Time.
DescrMetadata	N11 Bytes	Descriptive Metadata.

11.54 Motor Response

11.54.1 Definition

The response issued by a Wheel Motor to the Motion Actuation Subsystem informing about the execution of a Motor Command.

11.54.2 Functional Requirements

The Motor Response to a Motor Command includes:

1. Time the response is issued.
2. The Motor State represented by MotorStateOver, a number between 0 and 1 where:
    1. MotorStateOver=0, Wheel does not oppose to Torque applied by Motor
    2. MotorStateOver=1 Wheel operates correctly.
3. The Motor State represented by MotorStateUnder, a number between 0 and 1 where
    1. MotorStateUnder=0 Wheel is blocked
    2. MotorStateUnder = 1 Wheel operates correctly.

Real values represent intermediate states.

11.54.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/MotorResponse.json

11.54.4 Semantics

Label	Size	Description
Header	N1 Bytes	Wheel Response Header
- Standard	9 Bytes	The characters “CAV-MRR-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
MotorResponseID	N4 Bytes	Identifier Motor Response.
MotorID	N5 Bytes	Identifier of Motor.
MotorResponse	N7 Bytes	Set of Motor Response Data.
MotorResponseTime	17 Bytes	Time of the Motor Response is issued.
MotorState	N8 Bytes	State of Motor expressed as $0 \leq$ MotorStateOver ≤ 1 $0 \leq$ MotorStateUnder ≤ 1.
DescrMetadata	N10 Bytes	Descriptive Metadata

11.55 Spatial Data

11.55.1 Definition

Spatial Data is data produced by the Motion Actuation Subsystem regarding the Spatial Attitude of the CAV.

11.55.2 Functional Requirements

The unit of measure of Spatial Data are:

Name	Unit of measure
Odometer Data	Scalar whose coefficients are expressed in metres (m)
Speedometer Data	Scalar whose coefficients are expressed in metres/second (m/s)
Accelerometer Data	Scalar whose coefficients are expressed in metres/second/second (m/s^2)
Inclinometer Data	Vector of CAV inclination in the direction of travel and perpendicular to it, in degrees (°)

11.55.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/SpatialData.json

11.55.4 Semantics

Label	Size	Description
Header	N1 Bytes	Spatial Data Header
- Standard - SpatialData	9 Bytes	The characters “CAV-SPD-V”
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character “.”
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
SpatialDataID	N4 Bytes	ID of Spatial Data.
SpaceTime	N6 Bytes	Spatial Attitude and Time.
SpatialData	N7 Bytes	Set of Spatial Data.
- OdometerData	N8 Bytes	Data from Odometer.
- SpeedometerData	N9 Bytes	Data form Speedometer.
- AccelerometerData	N10 Bytes	Data from Accelerometer.
- InclinometerData	N11 Bytes	Data from Inclinometer.
DescrMetadata	N12 Bytes	Descriptive Metadata

11.56 Weather Data

11.56.1 Definition

Weather Data is a set of data that includes measures of:

1. Temperature
2. Humidity
3. Air Pressure
4. Ice
5. Wind
6. Condensed water in various states:

1. gaseous: fog
2. liquid: rain
3. frozen: snow, sleet, hail

11.56.2 Functional Requirements

1. Temperature measured in degrees °C.
2. Ice measured as Ice Conditions expressed as yes/no.
3. Wind measured as Wind Conditions expressed by Orientation and velocity measured in m/s.
4. Density of fog measured in meters of clear visibility.
5. Amount of rain measured in mm/h.
6. Hail measured in mm of hailstone size.
7. Snow measured mm/h.
8. Sleet measured in meters of clear visibility.

11.56.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/WeatherData.json

11.56.4 Semantics

Label	Size	Description
Header	N1 Bytes	Weather Data Header
- Standard-WeatherData	9 Bytes	The characters "CAV-WDT-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
WeatherDataID	N4 Bytes	
WeatherData	N5 Bytes	
- Temperature	N6 Bytes	Measured in °C
- RelativeHumidity	N7 Bytes	Measured in %
- Air pressure	N8 Bytes	Measured in millibars
- Ice	N10 Bytes	yes/no
- WindConditions	N11 Bytes	Azimuth and Elevation in degrees and velocity in m/s
- Fog	M12 Bytes	Measured in meters of clear visibility.
- Rain	N13 Bytes	Measured in mm/h
- Hail	N14 Bytes	Measured in mm size of hail.
- Snow	N15 Bytes	Measured in mm/hour.
- Sleet	N16 Bytes	Measured in metres of clear visibility.
DescrMetadata	N17 Bytes	Descriptive Metadata

11.57 Wheel Command

11.57.1 Definition

The command issued by the Motion Actuation Subsystem to rotate a Wheel.

11.57.2 Functional Requirements

A Wheel Command is expressed by:

1. Degrees representing the target angle of the wheel.
2. Seconds representing the Time the wheel should take to reach the target angle.

11.57.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/WheelCommand.json

11.57.4 Semantics

Label	Size	Description
Header	N1 Bytes	Wheel Command Header
- Standard	9 Bytes	The characters "CAV-WHC-V"
- Version	N2 Bytes	Major version – 1 or 2 Bytes
- Dot-separator	1 Byte	The character "."
- Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
SteeringID	N4 Bytes	Identifier of Steering.
SteeringCommands	N7 Bytes	Set of Steering Command.
- WheelCommandTime	N8 Bytes	Start and end Time of Wheel Command execution.
- Angle	N9 Bytes	Target angle of the wheel in degrees.
DescrMetadata	N11 Bytes	Descriptive Metadata

11.58 Wheel Response

11.58.1 Definition

The response issued by a Wheel informing about the execution of a Wheel Command.

11.58.2 Functional Requirements

The Response of a Wheel to a Wheel Command including:

1. Issue time.
2. Wheel State represented by a real $0 \leq$ and ≥ 1.
3. Final angle of the Wheel.

11.58.3 Syntax

https://schemas.mpai.community/CAV2/V1.0/data/WheelResponse.json

11.58.4 Semantics

Label	Size	Description
Header	N1 Bytes	Steering Response Header
– Standard	9 Bytes	The characters “CAV-WLR-V”
– Version	N2 Bytes	Major version – 1 or 2 Bytes
– Dot-separator	1 Byte	The character “.”
– Subversion	N3 Bytes	Minor version – 1 or 2 Bytes
WheelID	N4 Bytes	Identifier of a Wheel.
WheelResponseID	N5 Bytes	Identifier of Wheel Response.
WheelResponse	N6 Bytes	Wheel Response.
– WheelResponseTime	N7 Bytes	Time Wheel Response is issued.
– WheelState	N8 Bytes	State of Wheel represented by $0 \leq$ WheelState ≤ 1.
– WheelAngle	N9 Bytes	Angle reached by Wheel
DescrMetadata	N10 Bytes	Descriptive Metadata