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

MPAI Technical Specification

Multimodal Conversation
MPAI-MMC

WD 0.4

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 or any of its Members accept no responsibility whatsoever for damages or liability, direct or consequential, which may result from use of this Technical Specification.

Readers are invited to review Annex 1 – Notices and Disclaimers.

Multimodal Conversation

1 Introduction

Moving Picture, Audio and Data Coding by Artificial Intelligence (MPAI) is an international Standards Developing Organisation with the mission to develop *AI-enabled data coding standards*. Research has shown that data coding with AI-based technologies is generally *more efficient* than with existing technologies. Compression and feature-based description are notable examples of coding.

Conversation with Emotion (MPAI-MMC) is an MPAI Standard, comprising 5 Use Cases: "Conversation with Emotion", supporting audio-visual conversation with a machine impersonated by a synthetic voice and an animated face; "Multimodal Question Answering" supporting request for information about a displayed object; "Unidirectional Speech Translation", "Bidirectional Speech Translation" and "One-to-Many Speech Translation" supporting conversational translation application based on synthetic speech that preserves the speech features of the human.

The current version of MPAI-MMC has been developed by the MPAI Multimodal Conversation Development Committee (MMCC-DC). Future versions of the standard may extend the scope of the Use Cases and/or add new Use Cases in the scope of Multimodal Conversation.

In the following Terms beginning with a capital letter are defined in *Table 1*if they are specific to this MPAI-CUI Standard and to *Table 13Table 13* if they are common to all MPAI Standards.

The AI Framework (AIF) execution environment (MPA-AIF) [2] depicted in *Figure 1* enables Interoperable AI applications and services. Further details of the AIF Reference Model can be found in Annex 3.

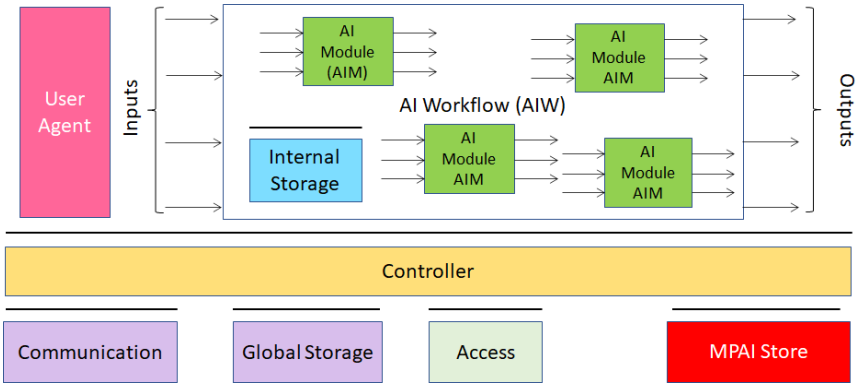

Figure 1 – The Components of the AI Framework (AIF)

The MPAI-MMC Application Standard normatively specifies 5 AI Frameworks (AIW) supporting the 5 MPAI-MMC identified Use Cases. The MPAI-MMC Use Case no. 3 "Unidirectional Speech Translation" interprets a Speech Segment uttered in a specified language to another specified languages preserving the characteristics of the original speech. *Figure 2* depicts an AIW standardised by MPAI-MMC, called "Unidirectional Speech Translation".

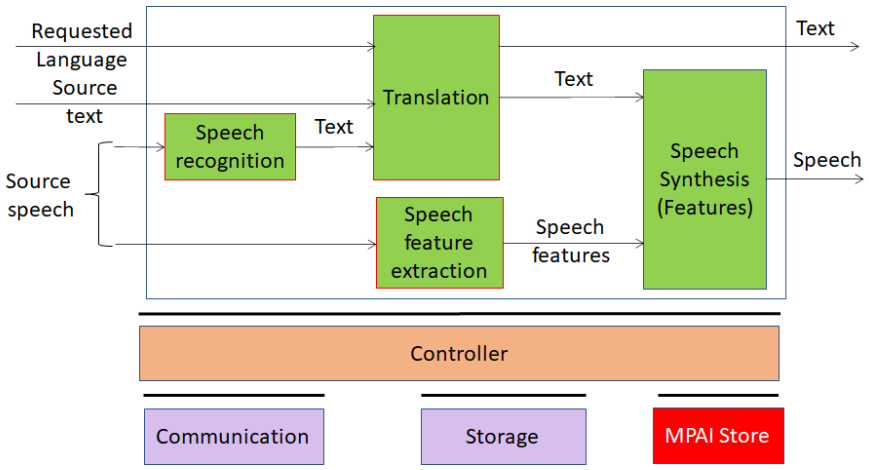

Figure 2 – An AIM example

MPAI-MMC normatively specifies the following aspects of an AIW:

1. The semantics and the format of the input data, e.g., "Requested Languages", "Source Text" and "Source Speech".
2. The function, e.g., "interpreting a Speech Segment or a Source Text from a language to another preserving the characteristics of the original Speech Segment".
3. The format of the output data, e.g., "Speech Segment" and "Text".

An AIW is composed of data processing elements – called AI Modules (AIM). **Error! Reference source not found.** and *Figure 4* depict two examples of the same AIM with the same function. **Error! Reference source not found.** includes the necessary knowledge (e.g., a neural network) and *Figure 4* accesses that knowledge from an external knowledge base.

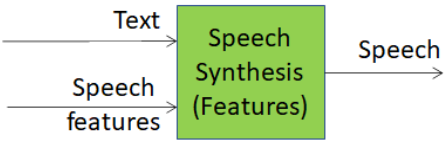

Figure 3 – An AIM with embedded knowledge

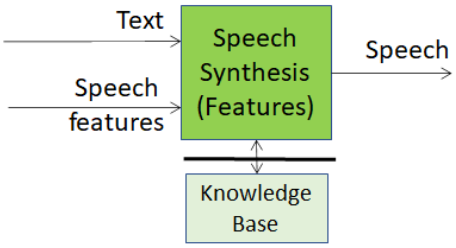

Figure 4 – An AIM with access to an external knowledge base

MPAI-MMC normatively specifies the following aspects of an AIM:

1. The format and semantics of the input data, e.g., "Text and Speech Features".
2. The function, e.g., "Produce a synthetic speech from text and Emotion Descriptors".
3. The format of the output data, e.g., "Speech Segment".

An AIM is defined by its function and interfaces, but not by its internal architecture, which may be based on AI (e.g., **Error! Reference source not found.**) or data processing (e.g., *Figure 4*), and implemented in software, hardware or hybrid software and hardware technologies.

MPAI normatively specifies the process, the tools and the data or the characteristics of the data to be used to Assess the Grade of Performance of an AIM or a AIW.

MPAI offers implementers 3 different Levels of compliance to MPAI Standards:

Level 1 An AIF Implementation running an AIW composed of AIMs performing any proprietary function and exposing any proprietary interface but exposing the interfaces required to be executed in the AIF.

Level 2 An AIF Implementation running an AIW composed of AIMs whose functions and interfaces are specified by an MPAI Application Standard.

Level 3 An Implementation running an AIW composed of AIMs certified to possess the attributes of Reliability, Robustness, Replicability and Fairness – collectively called Performance.

The MPAI ecosystem offers users access to the promised benefits of AI with a guarantee of increased transparency, trust and reliability as the Interoperability Level moves from 1 to 3. More informative details are provided by Annex 3.

2 Scope of Standard

The *Multimodal Conversation* Technical Specification (MPAI-MMC) includes 5 Use Cases sharing the characteristic of using AI to enable a form of human-machine conversation that emulates human-human conversation in completeness and intensity. They are: *Conversation with Emotion (CWE), Multimodal Question Answering (MQA), Unidirectional Speech Translation (UST), Bidirectional Speech Translation (BST) and One-to-many Speech Translation (OMT).* MPAI expects to produce future MPAI-MMC versions supporting enhanced current and new Use Cases.

This version of MPAI-MMC has been developed by the MMC-DC Development Committee

2.1 Conversation with Emotion (CWE)

When people talk, they use multiple modalities. Emotion is one of the key features to understand the meaning of the utterances made by the speaker. Therefore, a conversation system with the capability to recognize emotion can better understand the user and produce a better reply.

The Conversation with Emotion (MMC-CWE) Use Case handles conversation with emotion. It is a human-machine conversation system where the computer can recognize emotion in the user's speech and/or text, also using the video information of the face of the human to produce a reply coherent with the emotional state of the human.

2.2 Multimodal Question Answering (MQA)

Question Answering (QA) Systems answer a user's question presented in natural language. Current QA systems only deal with the case where input is in "text" form or "speech" form. However, there are cases where mixed inputs such as speech with an image are presented to the system. For example, a user asks a question: "Where can I buy this tool?" showing the picture of the tool. In Multimodal Question Answering (MMC-MQA) a machine responds to a question expressed in a text or in speech by a user showing an object using text and synthetic speech as output.

2.3 Unidirectional Speech Translation (UST)

In the Unidirectional Speech Translation (MMC-UST) Use Case, the system recognizes a voice uttered in a language by a speaker, converts the recognized voice into another language through automatic translation, and outputs a converted voice as text-type subtitles or as a synthesized voice preserving the speaker's features in the translated speech.

2.4 Bidirectional Speech Translation (BST)

In Bidirectional (as opposed to Unidirectional) Speech-to-Speech Translation (MMC-BST), two people converse, each speaking a different language. They may be in the same location, or they may be communicating remotely. The AIMs implementing the components may be implemented as online services, or they may be embedded on a single device. The flow of control (from speech recognition to text translation to speech synthesis) is identical to that of the Unidirectional case. The difference is that, rather than one such flow, two flows are provided – the first from language A to language B, and the second from B to A.

2.5 One-to-Many speech translation (OMT)

In One-to-Many (as opposed to Unidirectional or Bidirectional) Speech-to-Speech Translation (MMC-OMT), one person speaking his or her preferred language broadcasts to two or more audience members, each listening, and responding, in a different language. The speaker and audience may be in the same location, or the communication may be carried out remotely. The AIMs implementing the components may be implemented as online services, or they may be embedded on a single device. The flow of control (from speech recognition to text translation to speech synthesis) is identical to that of the Unidirectional case. However, rather than one such flow, multiple paired flows are provided – the first pair from language A to language B and B to A; the second from A to C and C to A; and so on.

2.6 Normative content of the Use Cases

Each Use Case normatively defines:

1. *AIW*: a structured aggregation of AIMs that implements the Use Case characterised by:
   a. The function performed by the AIW.
   b. The input and output data of the AIW.
   c. The topology and connections of the AIMs in the AIW.
2. *AI Modules*: processing elements that are characterised by:
   a. The function performed by the AIM.
   b. The input and output data of the AIM.
3. *Data formats*: any type of static (time independent) or dynamic (time dependent) data that is used as input and output of a AIW or an AIM.

The word *normatively* is to be interpreted to mean that if an implementer claims conformance to

1. a *AIW*, the implementation shall:
   a. Perform the AIW function specified in Chapter 5.
   b. All AIMs, their topology and connections should conform to the AIW Architecture specified in Chapter 5.
   c. The AIW and AIM input and output data should have the formats specified in Chapter 5.
2. an *AIM*, the implementation shall:
   a. Perform the AIM function specified by the appropriate section of Chapter 5.
   b. Receive as input and produce as output data in the formats specified in Chapter 5.
3. a *data format*, the data shall have the format specified in Section 0.

Users of this Technical Specification should note that:

1. This Technical Specification defines the possible levels of conformance but does not mandate any.
2. Implementers decide the level of conformance their implementation satisfies.
3. Implementers can use the Reference Software of this Technical Specification to develop their implementations.
4. The Conformance Testing specification can be used to test the conformity of an implementation to this Technical Specification.

5. Performance Assessors can assess the level of Performance of an implementation based on the Performance Assessment specification.

The Governance of the MPAI Ecosystem, outlined in Annex 2.

3 Terms and Definitions

The terms used in this standard whose first letter is capital have the meaning defined in *Table 1*.

Table 1 – Table of terms and definitions

Term	Definition
Emotion	An attribute that indicates an emotion out of a finite set of Emotions
Emotion Grade	The intensity of an Emotion
Emotion Recognition	An AIM that decides the final Emotion out of Emotions from different sources
Image analysis	An AIM that extracts features from video
Intention	Intention is the result of a question analysis that denotes information on the input question
Language Understanding	An AIM that analyses natural language as Text to produce its meaning and emotion included in the text
Meaning	Information extracted from the input text such as syntactic and semantic information
Question Analysis	An AIM that analyses the meaning of a question sentence and determines its Intention
Question Answering	An AIM that analyses the user's question and produces a reply based on the user's Intention
Speech Recognition	An AIM that converts speech to Text
Speech Synthesis	An AIM that converts Text or concept to speech
Text	A collection of characters drawn from a finite alphabet
Translation	An AIM that converts Text in a source language to Text in a target language

4 Normative References

This standard normatively references the following documents, both from MPAI and other standard organisations:

1. The governance of the MPAI ecosystem, N309
2. AI Framework Technical Specification, N293
3. ISO 639 – Codes for the Representation of Names of Languages — Part 1: Alpha-2 Code.
4. ISO/IEC 10646, Information technology – Universal Coded Character Set
5. …

5 Use Case Architectures

5.1 Conversation with Emotion (CWE)

5.1.1 Scope of Use Case

In the Conversation with Emotion (CWE) use case, a machine responds to a textual and/or vocal utterance made by a human in a way that is congruent with the human's utterance and emotional state as detected from the human's text and/or speech and face. The machine responds using text, synthetic speech and a face whose lips are animated by the synthetic speech.

5.1.2 Input/Output Data

The input and output data of this Use Case are:

Input	Comments
Text	Text typed by the human as additional information stream or as a replacement of the speech.
Speech	Speech of the human having a conversation with the machine.
Video	Video of the face of the human having a conversation with the machine.
Output	**Comments**
Text	Text of the speech produced by the machine.
Speech	Synthetic speech produced by the machine.
Video	Video of a face whose lips are animated by the speech produced by the machine.

5.1.3 Implementation Architecture

The operation of Conversation with Emotion develops in the following way:

1. Emotion is recognised in the following way and reflected in the speech production side.
    a. A set of emotion related cues are extracted from text, voice and video.
    b. Each text, speech and video recognition module recognises emotion independently.
    c. The Emotion recognition module fuses all emotions into the final emotion.
    d. The final emotion is transferred to the Dialog processing module.
2. The Dialog Processing module produces a reply based on the final emotion and meaning from the text and video analysis.
3. The Speech Synthesis (Emotion) module produces the speech from the reply in text with embedded emotion
4. The Face animation AIM produces the animated lips of a face consistently with the synthesised Speech drawing information from the Video of Faces Knowledge Base.

Figure 5 depicts the input/output data, the AIMs and the data exchanged between AIMs.

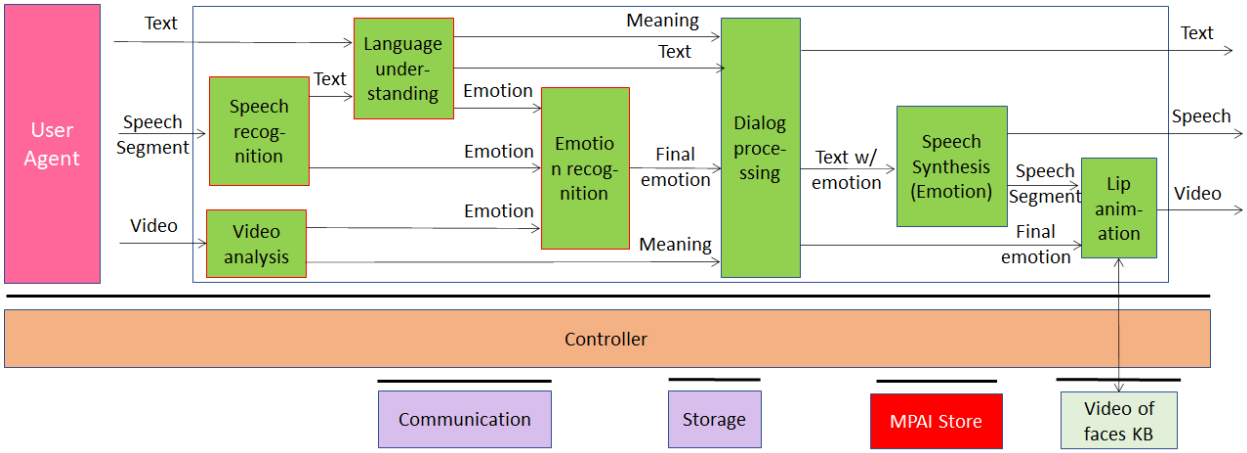

Figure 5 – Architecture of Conversation with Emotion

5.1.4 AI Modules

The AI Modules of Conversation with Emotion perform the functions described in *Table 2*.

Table 2 – AI Modules of Conversation with Emotion

AIM	Function

Language understanding	Analyses natural language in a text format to produce its meaning and emotion included in the text.
Speech Recognition	Analyses the voice input and generates text output and emotion carried by it.
Video analysis	Analyses the video and recognises the emotion it carries.
Emotion recognition	Determines the final emotion from multi-source emotions.
Dialog processing	Analyses user's Meaning and produces Reply based on the meaning and emotion implied by the user's text.
Speech synthesis	Produces speech from Reply (the input text).
Lips animation	Produces a video of a face whose lips are animated consistently with the synthesised Speech.

5.1.5 AIW Metadata

Specified in Annex 4 Section 2.

5.2 Multimodal Question Answering (MQA)

5.2.1 Scope of standard

A human asks a question in natural language expressed as text and/or speech while showing an object the question refers to. The machine responds to the question in text and synthetic speech.

5.2.2 Input/output data

Input	**Comments**
Text	Text typed by the human as additional information stream or as a replacement of the speech.
Speech	Speech of the human asking a question to the machine.
Video	Video of the human showing an object in their hands.
Output	**Comments**
Text	Text of the speech produced by the machine.
Speech	Synthetic speech produced by the machine.

5.2.3 Implementation Architecture

The operation of Multimodal Question Answering develops in the following way:

1. A question is asked in the form of text or voice.
2. The meaning of the question is recognised.
3. Video analysis identifies the object and sends it to Language Understanding.
4. Language Understanding fuses the multimodal inputs and generates the integrated meaning.
5. Intention Analysis determines the Intention of the question and sends it to QA.
6. Question Answering uses the intention of the question and the Meaning to produce the answer.
7. Speech Synthesis (Text) produces the speech from the answer in text.

Figure 6 depicts the input/output data, the AIMs and the data exchanged between AIMs.

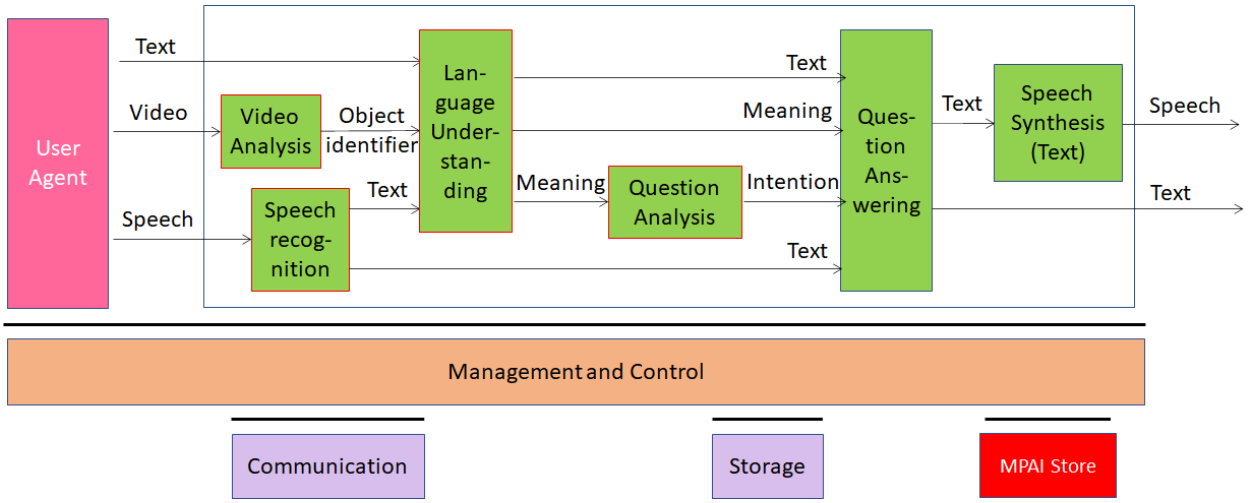

Figure 6 – Architecture of Multimodal Question Answering

5.2.4 AI Modules

The AI Modules of Multimodal Question Answering are given in *Table 3*.

Table 3 – AI Modules of Multimodal Question Answering

AIM	Function
Language understan-ding	Analyses natural language expressed as text to produce the meaning of the text.
Speech Recognition	Analyses the speech input and generates text output.
Speech synthesis	Converts input text to speech.
Video analysis	Analyses video and produces the name of object in focus.
Question analysis	Analyses the Meaning of the sentence and determines the Intention .
Question Answering	Analyses user's question and produces a Reply.

5.2.5 AIW Metadata

Specified in Annex 5 Section 2.

5.3 Unidirectional Speech Translation (UST)

5.3.1 Scope of Use Case

In Unidirectional Speech Translation, spoken segments in Language A are translated into spoken segments in Language B.. The AIMs implementing the components may be implemented as online services, or they may be embedded on a single device. The flow of control is from speech recognition to text translation and then, to speech synthesis.

5.3.2 Input/output data

Input	**Comments**
Desired languages	User-specified input and output languages
Speech	Speech produced by a human desiring spoken translation in the specified language.
Text	Alternative textual source information to be translated to the specified language.
Output	**Comments**
Speech	Translated speech.
Text	Text of the translated speech.

5.3.3 Implementation Architecture

Figure 7 describes the input/output data, the AIMs and the data exchanged between AIMs.

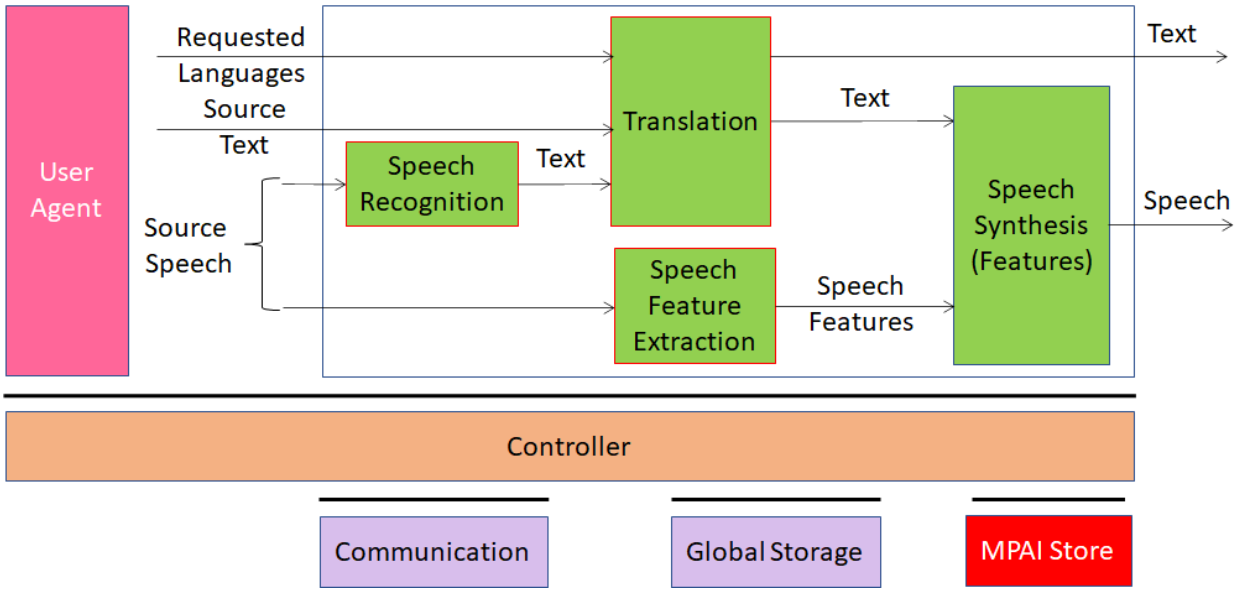

Figure 7 – Architecture of Unidirectional Speech Translation

5.3.4 AI Modules

The AI Modules of Unidirectional Speech Translation are given in *Table 4*.

Table 4 – AI Modules of Unidirectional Speech Translation

AIM	Function
Speech Recognition	Converts Speech into Text.
Translation	Translates the user text input in source language to the target language.
Speech feature extraction	Extracts Speech features such as tones, intonation, intensity, pitch, emotion, intensity or speed from the input voice specific of the speaker.
Speech Synthesis (Features)	Produces Speech from the text resulting from translation with the speech features extracted from the speaker of the source language

5.3.5 AIW Metadata

Specified in Annex 6 Section 2.

5.4 Bidirectional Speech Translation (BST)

5.4.1 Scope of Use Case

In Bidirectional (as opposed to Unidirectional) Speech Translation, two people converse, each speaking a different language. They may be in the same location, or they may be communicating remotely. The AIMs implementing the components may be implemented as online services, or they may be embedded on a single device. The flow of control (from speech recognition to text translation to speech synthesis) is identical to that of the Unidirectional case. The difference is that, rather than one such flow, two flows are provided – the first from language A to language B, and the second from B to A.

5.4.2 Input/output data

Input	Comments
Desired languages	User-specified input and output languages
Speech1	Speech produced by human1 desiring spoken translation in the specified language.
Text1	Alternative textual source information1 to be translated to the specified language.
Speech2	Speech produced by human2 desiring spoken translation in the specified language.
Text2	Alternative textual source information2 to be translated to the specified language.
Output	**Comments**
Speech1	Translated speech of Speaker 1.
Text1	Text of the translated speech by Speaker 1.
Speech2	Translated speech of Speaker 2.
Text2	Text of the translated speech by Speaker 2.

5.4.3 Implementation Architecture

Figure 8 depicts the AIMs and the data exchanged between AIMs.

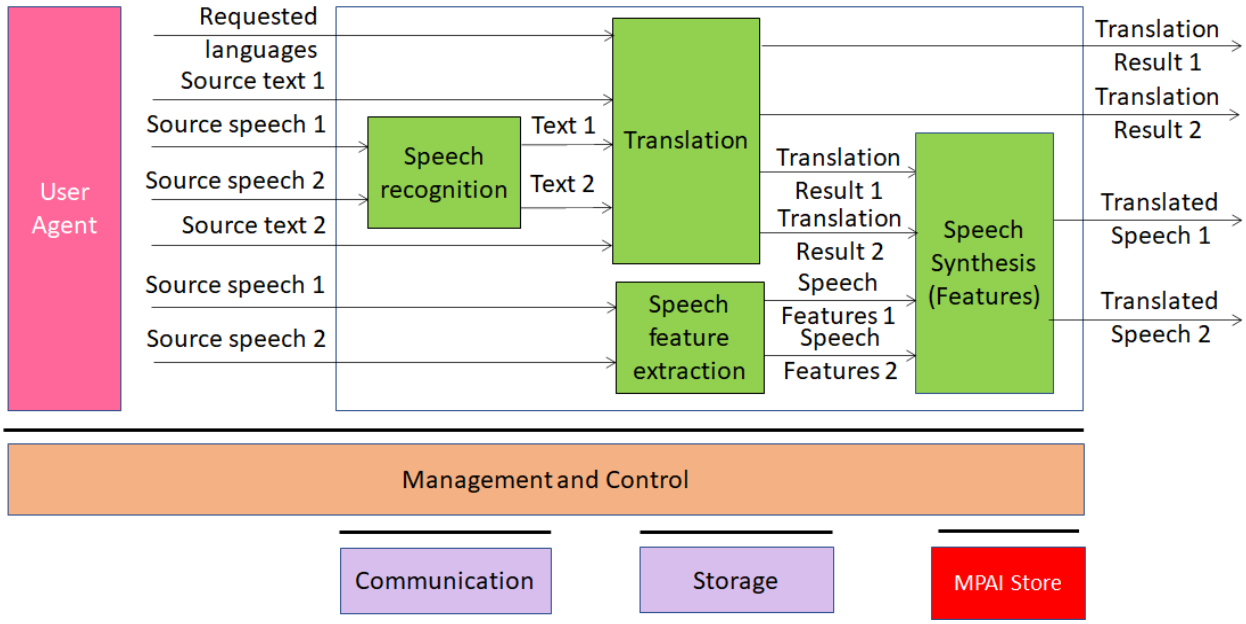

Figure 8 – Architecture of Bidirectional Speech Translation

5.4.4 AI Modules

The AI Modules are given in *Table 5*.

Table 5 – AI Modules of Bidirectional Speech Translation

AIM	Function
Speech Recognition	Converts 2 independent Speech inputs into 2 independent Text outputs.
Translation	Translates 2 independent Text inputs in two independent Text outputs.
Speech feature extraction	Extracts 2 independent Speech features from the 2 input Speeches.
Speech Synthesis (Features)	Produces 2 Speeches from the texts resulting from translation with the speech features extracted from the corresponding speaker.

5.4.5 AIW Metadata

Specified in Annex 7 Section 2.

5.5 One-to-Many speech translation (OMT)

5.5.1 Scope of Use Case

In One-to-Many (as opposed to Unidirectional or Bidirectional) Speech Translation, one person speaking his or her preferred language broadcasts to two or more audience members, each listening, and potentially responding, in a different language. The speaker and audience may be in the same location, or the communication may be carried out remotely. The AIMs implementing the components may be implemented as online services, or they may be embedded on a single device. The flow of control (from speech recognition to text translation to speech synthesis) is identical to that of the Unidirectional case. However, rather than one such flow, multiple paired flows are provided – the first pair from language A to language B and B to A; the second from A to C and C to A; and so on.

5.5.2 Input/output data

Input	Comments
Desired languages	User-specified input and output languages
Speech	Speech produced by human desiring spoken and text translation in a specified set of languages.
Text	Alternative textual source information to be translated to the specified set of languages.
Output	**Comments**
Speech	Translated speech speech segments.
Text1	Text of the translated speech segments.

5.5.3 Implementation Architecture

Figure 9 depicts the AIMs and the data exchanged between AIMs.

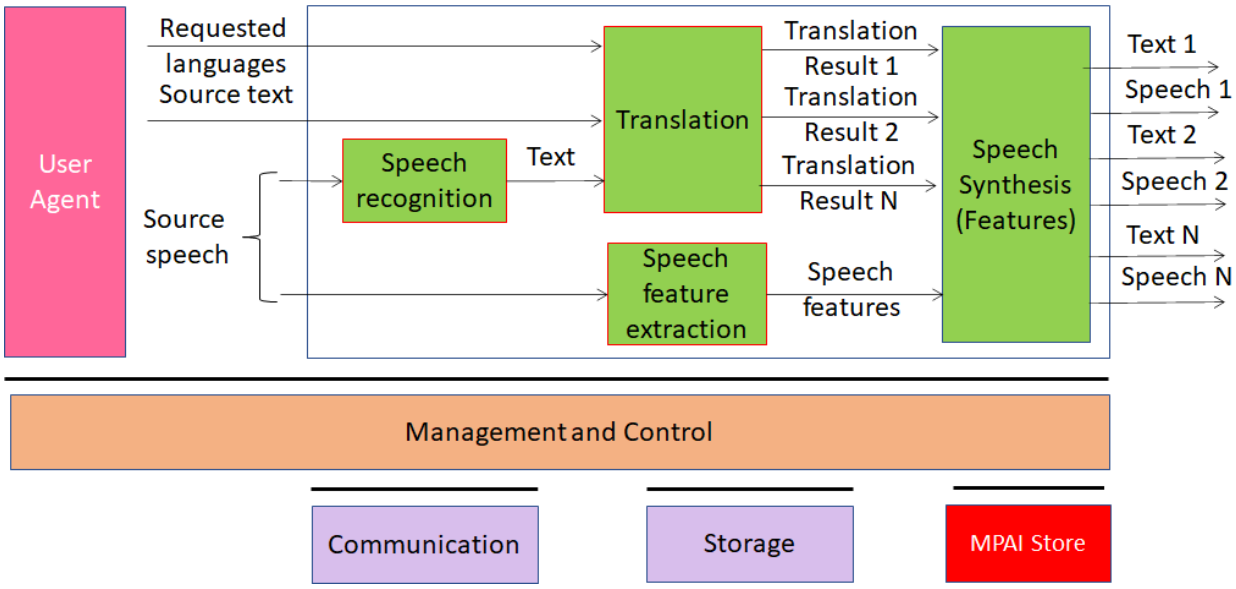

Figure 9 – Architecture of One-to-Many Speech Translation (OMT)

5.5.4 AI Modules

The AI Modules of Personalized Automatic Speech Translation are given in *Table 6*.

Table 6 – AI Modules of One-to-Many Speech Translation

AIM	Function
Speech Recognition	Converts 1 Speech input into a set of Text outputs of the specified languages
Translation	Translates 1 Text input into a set of Text of the Requested Languages
Speech feature extraction	Extracts Speech Features from the input voice specific of the speaker.
Speech Synthesis (Features)	Uses the set of Text resulting from translation and the Speech Features extracted from the speaker of the Source Language to produce a set of Speech Segments in the specified languages.

5.5.5 AIW Metadata

Specified in Annex 8 Section 2.

6 AI Modules

This Chapter specifies the AIMs and their input and output data employed by all Use Cases specified in this Standard.

Section 6.1 lists the AIMs and their data in tabular form and using the AIM Metadata specified by the AI Framework (MPAI-AIF) Standard.

Section 0 specifies the formats of the input and output data used in this Standard.

The reader is alerted that some data formats in this Standard are shared with the Context-based Audio Enhancement (MPAI-CAE) Standard. The specification of such data formats is repeated verbatim in both standards. MPAI plans on creating a future specification that will contain all data formats that are shared by more than one MPAI Standard.

6.1 MPAI-MMC AIMs and their data

6.1.1 Conversation with Emotion (CWE)

The AIMs and the data formats used by the Conversation with Emotion Use Case are given by *Table 7*.

Table 7 – Conversation with Emotion AIMs and data formats

AIM	Input Data	Output Data
Video analysis	Video	Emotion Meaning
Speech recognition	Input Speech	Text Emotion
Language understanding	Input Text Recognised Text	Text Emotion Meaning
Emotion recognition	Emotion (from text) Emotion (from speech) Emotion (from image)	Final Emotion
Dialog processing	Text	Text with Emotion

	Meaning (text/speech) Final Emotion Meaning (video)	Text
Speech Synthesis (Emotion)	Text with Final Emotion	Speech
Lips animation	Synthesized Speech Final Emotion Response of Face video KB	Video Query Face video KB

The AIM Metadata are given in Annex 4 Section 3.

6.1.2 Multimodal Question Answering (MQA)

The AIMs and the data formats used by the Multimodal Question Answering Use Case are given by *Table 8*.

Table 8 – Multimodal Question Answering AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Speech	Text
Image analysis	Image	Text
Language understanding	Text Text	Meaning Meaning
Question analysis	Meaning	Intention
Question Answering	Text Meaning Intention	Text
Speech Synthesis (Text)	Text	Speech

The AIM Metadata are given in Annex 5 Section 3.

6.1.3 Unidirectional Speech Translation (UST)

The AIMs and the data formats used by the Unidirectional Speech Translation Use Case are given by *Table 9*.

Table 9 – Unidirectional Speech Translation AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Speech	Text
Translation	Text Requested language	Text (Translation result)
Speech feature extraction	Speech	Speech features
Speech synthesis (Features)	Text (Translation result) Speech features	Speech

The AIM Metadata are given in Annex 6 Section 3.

6.1.4 Bidirectional Speech Translation (BST)

The AIMs and the data formats used by the Bidirectional Speech Translation Use Case are given by *Table 10*.

Table 10 – Bidirectional Speech Translation AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Input speech	Text
Translation	Requested language Text	Text (Translation result)
Speech feature extraction	Speech	Speech features
Speech synthesis (Features)	Text (Translation result) Speech features	Output speech

The AIM Metadata are given in Annex 7 Section 3.

6.1.5 One-to-many Speech Translation (OMT)

The AIMs and the data formats used by the One-to-many Speech Translation Use Case are given by *Table 11*.

Table 11 – One-to-many Speech Translation AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Digital Speech	Text
Translation	Requested languages Text Speech	Text (Translation result)
Speech feature extraction	Digital speech	Speech features
Speech synthesis (Features)	Text (Translation result) Speech features	Digital speech Text (Translation result)

The AIM Metadata are given in Annex 4 Section 8.

6.2 Data Formats

Table 12 lists all data formats specified in this Technical Specification. The first column gives the name of the data format, the second the subsection where the data format is specified and the third the Use Case(s) making use of it.

Table 12 – Data formats

Name of Data Format	Subsection	Use Case
Text	6.2.1	CWE
		MQA
		UST
		BST
		OMT
Speech	6.2.2	CWE
		MQA
		UST
		BST
		OMT
Video	6.2.3	CWE
Emotion	6.2.4	CWE
Text with emotion	6.2.5	CWE

Video of faces KB Query Format	0	CWE
Object identifier	6.2.7	MQA
Meaning	0	CWE
		MQA
Intention	0	MQA
Language identifier	6.2.10	UST
		BST
		OMT
Speech features	6.2.11	UST
		BST
		OMT

6.2.1 Text

Encoded according to ISO/IEC 10646, Information technology – Universal Coded Character Set (UCS) to support most languages in use.

6.2.2 Speech Segment

Speech Segment ia a .wav file of the digital representation of analogue speech sampled in the 8-96 kHz frequency range and with 16-24 bits/sample (linear).

6.2.3 Video

Video satisfies the following specifications:

1. Pixel shape: square
2. Bit depth: 8-10 bits/pixel
3. Aspect ratio: 4/3 and 16/9
4. 640 < # of horizontal pixels < 1920
5. 480 < # of vertical pixels < 1080
6. Frame frequency 50-120 Hz
7. Scanning: progressive
8. Colorimetry: ITU-R BT709 and BT2020
9. Colour format: RGB and YUV
10. Compression: uncompressed; if compressed AVC, HEVC

6.2.4 Emotion

Human Emotion is represented by.

```
{
   "$schema": "http://json-schema.org/draft-07/schema",
   "definitions": {
      "EmotionType": {
         "type": "object",
         "properties": {
            "emotionDegree": {"type": "integer"},
            "emotionName": {"type": "string"},
            "emotionSetName": {"type": "string"},
          }
"type": "object",
   "properties": {
      "primary": {"$ref": "#/definitions/EmotionType"},
      "secondary": {"$ref": "#/definitions/EmotionType"}
   }
```

}

Semantics of emotion

Name	*Definition*
`emotionType`	Describes the emotion that the input carries.
`emotionDegree`	Describes the degree of the emotion expressed in number.
`emotionName`	Describes the name of the emotion.
`emotionSetName`	Name of the emotion set which is used for describing the final emotion. MPAI emotion set is used as a baseline and other sets are possible.

The following list of emotions suitable for vocal expression seems to us reasonable, but is offered without theoretical or research-based commitment. It has been collected and sorted from several sources, some of them linked below under References.

Emotions are expressed vocally through combinations of prosody (pitch, rhythm, and volume variations); separable speech effects (such as degrees of voice tension, breathiness, etc.; see Effects, below); and vocal gestures (laughs, sobs, etc.).

Emotions can of course be combined: one can be both sad and angry. For that reason and others, no list of emotion names can be definitive, just as no listing of color names could be final. Accordingly, we suggest that a mechanism be defined whereby implementors of specific use cases or AIMs (modules) for emotional speech can register with MPAI the set of emotions that these offer to cover or enable, including unique names or identifiers. Registration procedures per implementation should be designed by the MPAI authorities concerned with AIM and use case registration in general, with consultations of the interested parties.

Basic Emotions originally following Paul Eckman [reference]

LEVEL 1	LEVEL 2	LEVEL 3
HAPPINESS	happy	joyful content delighted amused
SADNESS	sad	lonely grief-stricken discouraged depressed disappointed

CALM	calm	peaceful/serene resigned
FEAR	fearful/scared	terrified anxious/uneasy
ANGER	anger	furious irritated frustrated
DISGUST	disgust	loathing
SOCIAL DOMINANCE, CONFIDENCE	arrogant confident submissive	
PRIDE/SHAME	proud ashamed	arrogant guilty/remorseful/sorry embarrassed
HURT	hurt jealous	
APPROVAL, DISAPPROVAL	admiring/approving disapproving indifferent	awed contemptuous
SURPRISE	surprised	astounded startled
ATTENTION	attentive	expectant/anticipating thoughtful distracted/absent-minded vigilant hopeful/optimistic
INTEREST	interested	fascinated curious bored
UNDERSTANDING	comprehending	uncomprehending bewildered/puzzled
BELIEF	credulous	skeptical
AROUSAL	aroused/excited/energetic	cheerful playful

		lethargic sleepy

Semantics

Emotion	Meaning
admiring/approving	emotion due to perception that others' actions or results are valuable
amused	positive emotion combined with interest (cognitive)
anger	emotion due to perception of physical or emotional damage or threat
anxious/uneasy	low or medium degree of fear, often continuing rather than instant
aroused/excited/energetic	cognitive state of alertness and energy
arrogant	emotion communicating social dominance
arrogant	high degree of pride, often offensive to others
astounded	high degree of surprised
attentive	cognitive state of paying attention
awed	approval combined with incomprehension or fear
bewildered/puzzled	high degree of incomprehension
bored	not interested
calm	relative lack of emotion
cheerful	energetic combined with and communicating happiness
comprehending	cognitive state of successful application of mental models to a situation
confident	emotion due to belief in ability
contemptuous	high degree of disapproval
content	medium or low degree of happiness, continuing rather than instant
credulous	cognitive state of conformance to mental models of a situation
curious	interest due to drive to know or understand
delighted	high degree of happiness, often combined with surprise
depressed	high degree of sadness, continuing rather than instant, combined with lethargy (see AROUSAL)
disappointed	sadness due to failure of desired outcome
disapproving	not approving
discouraged	sadness combined with frustration

disgust	emotion due to urge to avoid, often due to unpleasant perception or disapproval
distracted/absent-minded	not attentive to present situation due to competing thoughts
embarrassed	shame due to consciousness of violation of social conventions
expectant/anticipating	attentive to (expecting) future event or events
fascinated	high degree of interest
fearful/scared	emotion due to anticipation of physical or emotional pain or other undesired event or events
frustrated	angry due to failure of desired outcome
furious	high degree of anger
grief-stricken	sadness due to loss of an important social contact
guilty/remorseful/sorry	shame due to consciousness of hurting or damaging others
Happy	positive emotion, often continuing rather than instant
hopeful/optimistic	expectation of good outcomes
hurt	emotion due to perception that others have caused social pain or embarrassment
indifferent	neither approving nor disapproving
interested	cognitive state of attentiveness due to salience or appeal to emotions or drives
irritated	low or medium degree of anger
jealous	emotion due to perception that others are more fortunate or successful
joyful	high degree of happiness, often due to a specific event
lethargic	not aroused
loathing	high degree of disgust
lonely	sadness due to insufficient social contact
peaceful/serene	calm combined with low degree of happiness
playful	energetic and communicating willingness to play
proud	emotion due to perception of positive social standing
resigned	calm due to acceptance of failure of desired outcome, often combined with low degree of sadness
sad	negative emotion, often continuing rather than instant, often associated with a specific event
skeptical	not credulous
sleepy	not aroused due to need for sleep
startled	surprised by a sudden event or perception

submissive	emotion communicating lack of social dominance
surprised	cognitive state due to violation of expectation
terrified	high degree of fear
thoughtful	attentive to thoughts
uncomprehending	not comprehending
vigilant	high degree of expectation or attentiveness

6.2.5 Text with Emotion

Text With Emotion is represented as follows.

6.2.5.1 Syntax

```
{
   "$schema": "http://json-schema.org/draft-07/schema",
   "definitions": {
      "TextWithEmotionType": {
         "type": "object",
         "properties": {
            "text": {“type”: "string"},
            "emotionDegree": {"type": "integer"},
            "emotionName": {"type": "string"},
            "emotionSetName": {"type": "string"},
          }
"type": "object",
   "properties": {
      "primary": {"$ref": "#/definitions/TextWithEmotionType"},
      "secondary": {"$ref": "#/definitions/TextWithEmotionType"}
   }
}
```

6.2.5.2 Semantics

Semantics of text with emotion

Name	*Definition*
`TextWithEmotionType`	Describes the emotion that the text carries.
`emotionDegree`	Describes the degree of the emotion expressed in number.
`emotionName`	Describes the name of the emotion.
`emotionSetName`	Name of the emotion set which is used for describing the final emotion. MPAI emotion set is used as a baseline and other sets are possible.

6.2.6 Video of faces KB Query Format

The Video of faces KB is queried with an Emotion. The response is a Video of a human face. All faces must be aligned.

6.2.7 Object identifier

An object is identified as follows.

6.2.7.1 Syntax

```
{
   "$schema": "http://json-schema.org/draft-07/schema",
   "definitions": {
      "objectIdentifier": {
         "type": "object",
         "properties": {
            "objectImageLabel": {“type”: "string"},
            "confidenceLevel": {"type": "integer"},
          }
"type": "object",
   "properties": {
      "primary": {"$ref": "#/definitions/ObjectIdentifierType"},
      "secondary": {"$ref": "#/definitions/ObjectIdentifierType"}
   }
}
```

6.2.7.2 Semantics

Name	*Definition*
`objectIdentifier`	Tool for describing the output of the “Video analysis AIM”.
`objectImageLabel`	Describes the recognized object’s label in the object image.
`confidenceLevel`	Describes the confidence level of the object image label recognized by the “Video analysis”.

6.2.8 Meaning

This subclause specifies data formats to describe meaning which is the outputs of Language Understanding AIM. The “meaning” consists of the following elements.

- POS_tagging
- NE_tagging
- Dependency_tagging
- SRL_tagging

6.2.8.1 Syntax

```
{
```

```
  "$schema": "http://json-schema.org/draft-07/schema",
  "definitions": {
    "meaning": {
      "type": "object",
      "properties": {
        "POS_tagging": {
          "POS_tagging_set": {“type”: "string"},
          " POS_tagging_result": {“type”: "string"}
        },
       "NE_tagging": {
          "NE_tagging_set": {“type”: "string"},
          " NE_tagging_result": {“type”: "string"}
        }
       "dependency_tagging": {
          "dependency_tagging_set": {“type”: "string"},
          "dependency_tagging_result": {“type”: "string"}
      }
       "SRL_tagging": {
          " SRL_tagging_set": {“type”: "string"},
          " SRL_tagging_result": {“type”: "string"}
      }
    }
  },
  "type": "object",
  "properties": {
    "primary": { "$ref": "#/definitions/meaning" },
    "secondary": { "$ref": "#/definitions/meaning" }
  }
}
```

6.2.8.2 Semantics

Name	*Definition*
`Meaning`	Provides an abstract of description of Language analysis results, which can be done in Language Understanding AIM.
`POS_tagging`	Describes POS tagging results including information on the POS tagging set and tagged results of the User question. POS: Part of Speech such as noun, verb, etc.
`NE_tagging`	Describes NE tagging results including information on the NE tagging set and tagged results of the User question. NE: Named Entity such as Person, Organization, Fruit, etc.
`dependency_tagging`	Describes dependency tagging results including information on the dependency tagging set and tagged results of the User question. Dependency indicates the structure of the sentence such as subject, object, head of the relation, etc.
`SRL_tagging`	Describes SRL(Semantic Role Labelling) tagging results including information on the SRL tagging set and tagged results of the User question. SRL indicates the semantic structure of the sentence such as agent, location, patient role, etc.

6.2.9 Intention

This subclause specifies data formats to describe intention which is the outputs of Question analysis AIM. The "intention" consists of the following elements.

- qtopic
- qfocus
- qLAT
- qSAT

6.2.9.1 Syntax

```
{
   "$schema": "http://json-schema.org/draft-07/schema",
   "definitions": {
      "Intention": {
         "type": "object",
         "properties": {
            "qtopic": {"type": "string"},
            "qfocus": {"type": "string"},
            "qLAT": {"type": "string"},
            "qSAT": {"type": "string"},
            "qdomain": {"type": "string"},
          }
"type": "object",
   "properties": {
      "primary": { "$ref": "#/definitions/intention" },
      "secondary": { "$ref": "#/definitions/intention" }
   }
}
```

6.2.9.2 Semantics

Name	*Definition*
`Intention`	Provides abstracts of Intention of User Question description. Intention of User Question is sent to QA AIM for providing answers to the user.
`qtopic`	Describes topic of the question. Question topic is the object or event that the question is about. Ex. Qtopic is King Lear in "Who is the author of King Lear?".
`qfocus`	Describes the focus of the question, which is the part of the question that, if replaced by the answer, makes the question a stand-alone statement. Ex. What, where, who, what policy. Which river, etc. Example. Question: Who is the president of USA? (The word "Who" is the focus of the question and it will be replaced by "Bydon" in the Answer.) Answer: Bydon is the president of USA.

Name	*Definition*
qLAT	Describes the lexical answer type of the question.
qSAT	Describes the semantic answer type of the question. QSAT corresponds to Named Entity type of the language analysis results.
qdomain	Describes the domain of the question such as “science”, “weather”, “history”. Ex. Who is the third king of Yi dynasty in Korea? (qdomain: history)

The following example shows the question analysis result of the user’s question, “Who is the author of King Lear?” The question analysis result in the example shows that the domain of the question is “Literature”, the topic of the question is “King Lear”, the focus of the question is “Who”.

```
"intention": [
                {
                        "qdomain": "Literature",
                        "qtopic": " King Lear ",
                        "qfocus": " who ",
                        "qLAT": " author ",
                        "qSAT": " person ",
                }
]
```

The following example shows the result of the analysed question of “How do you make Kimchi?” The question analysis result in the example shows that the domain of the question is “Cooking”, the topic of the question is “Kimchi”, the focus of the question is “how”.

```
"intention": [
                {
                        "qdomain": "Cooking",
                        "qtopic": " Kimchi",
                        "qfocus": " How ",
                        "qLAT": " cooking method ",
                        "qSAT": " method ",
                }
]
```

6.2.10 Language identifier

Represented as specified by ISO 639 – Codes for the Representation of Names of Languages — Part 1: Alpha-2 Code.

6.2.11 Speech Features

Speech Features are digitally represented as follows.

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

```
        "type": "object",
        "properties": {
           "pitch": {“type”: "integer"},
           "tone": {"type": "string"},
           "intonation": {"type": "string"},
           "intensity": {"type": "string"},
           "speed": {"type": "string"},
           "emotion": {"type": "EmotionType"},
           "NNspeechFeatures": {"type": "vector of floating point"},
        }
"type": "object",
  "properties": {
     "primary": { "$ref": "#/definitions/SpeechFeatureType" },
     "secondary": { "$ref": "#/definitions/SpeechFeatureType" }
  }
}
```

6.2.11.1 Semantics

Name	*Definition*
SpeechFeatures	Describes speech features extracted from the input speech.
SpeechFeatureType	Describes type of the speech features extracted from the input speech.
NNSpeechFeatures	Describes speech features extracted from the input speech by Neural Network
pitch	Describes perceived tone frequency of a sound. Pitch is the quality that makes it possible to judge sounds as "higher" and "lower".
tone	Tone is a variation in the pitch of the voice while speaking.
intonation	Intonation, in phonetics, the melodic pattern of an utterance. Intonation is primarily a matter of variation in the pitch level of the voice, but in such languages as English, stress and rhythm are also involved. Intonation conveys differences of expressive meaning (e.g., surprise, anger, wariness).
intensity	Describes loudness of a speech which is subjective perception of sound pressure.
speed	Describes speech tempo or speech rate which a measure of the number of speech units of a given type produced within a given amount of time.
emotion	Describes the emotion that the input speech carries.

Name	*Definition*
emotiontype	Describes the type of emotion that the input speech carries.

7 References

The references provided here are for information purpose.

1. [1] Ekman, Paul (1999), "Basic Emotions", in Dalgleish, T; Power, M (eds.), Handbook of Cognition and Emotion (PDF), Sussex, UK: John Wiley & Sons

Annex 1 – MPAI-wide terms and definitions (Normative)

The Terms used in this standard whose first letter is capital and are not already included in *Table 1* are defined in *Table 13*.

Table 13 – MPAI-wide Terms

Term	Definition
AI Framework (AIF)	The environment where AIWs are executed.
AI AIW (AIW)	An organised aggregation of AIMs implementing a Use Case receiving AIM-specific Inputs and producing AIM-specific Outputs according to its Function.
AI Module (AIM)	A processing element receiving AIM-specific Inputs and producing AIM-specific Outputs according to according to its Function.
Application	A usage domain target of an Application Standard
Conformance	The attribute of an Implementation of being a correct technical Implementation of a Technical Specification.
Conformance Tester	An entity authorised by MPAI to Test the Conformance of an Implementation.
Conformance Testing	The normative document specifying the procedures, the tools, the data sets and/or the data set characteristics to Test the Conformance of an Implementation.
Conformance Testing Means	Procedures, tools, data sets and/or data set characteristics to Test the Conformance of an Implementation.
Data format	The standard digital representation of data and their semantics.
Ecosystem	The ensemble of MPAI, MPAI Store, Implementers, Conformance Testers, Performance Testers and Users of MPAI-AIF Implementations as needed to enable an Interoperability Level.
Explainability	The ability to trace the output of an implementation back to the inputs that have produced it.
Fairness	The attribute of an implementation whose extent of applicability can be assessed by making the training set and/or network open to testing for bias and unanticipated results.
Function	The expected result of a AIW of an AIM on input data.
Identifier	A name that uniquely identifies an Implementation.
Implementation	An embodiment of: 1. The MPAI-AIF Technical Specification. 2. A AIW/AIM of a particular Level (1-2-3) from a Use Case of an Application Standard.
Interoperability	The possibility of an AIM Implementation to be functionally replaced by another AIM Implementation having the same Interoperability Level:
Interoperability Level	One of the following: Level 1 AIM Implementations are proprietary but their AIWs can be executed in an AIF Implementations. Level 2 AIM Implementations Conform to the Conformance Testing of an Application Technical Specification.

	Level 3 AIM Implementations Perform according to the Performance Testing of an Application Technical Specification.
Normativity	The set of attributes of a technology or a set of technologies specified by the applicable parts of an MPAI standard.
Performance	The attribute of an Implementation of being Reliable, Robust, Fair and Replicable.
Performance Assessment	The normative document specifying the procedures, the tools, the data sets and/or the data set characteristics to Assess the Grade of Performance of an Implementation.
Performance Assessment Means	Procedures, tools, data sets and/or data set characteristics to Assess the Performance of an Implementation.
Performance Assessor	An entity authorised by MPAI to Assess the Performance of an Implementation in a given Application domain
Profile	A particular subset of the technologies that are used in MPAI-AIF or a Use Case and, where applicable, the classes, other subsets, options and parameters relevant to that subset.
Reference Software	A technically correct software implementation of a Technical Specification containing source code, or source and compiled code.
Reliability	The attribute of an Implementation that performs as specified by the standard, profile and version the Implementation refers to, e.g., within the application scope, stated limitations, and for the period of time specified by the Implementer.
Replicability	The attribute of an Implementation whose Performance, as Assessed by a Performer, can be replicated, within an agreed level, by another Performer.
Robustness	The attribute of an Implementation that copes with data outside of the stated application scope with an estimated degree of confidence.
Service Provider	An entrepreneur who offers an Implementation as a service (e.g., a recommendation service) to Users.
Standard	The ensemble of Technical Specification, Reference Software, Conformance Testing and Performance Assessment of an MPAI application Standard. MPAI-AIF does not include Performance Assessment.
Technical Specification	(Framework) the normative specification of the AI Framework. (Application) the normative specification of the set of Use Cases belonging to an Application Domain along with the AIMs required to Implement the Use Cases. the collection of Use Cases relevant to the Application Domain that include: 1. The formats of the Input/Output data of the AIWs implementing the Use Cases. 2. The Topology and Connections of the AIMs of the Use Cases. 3. The formats of the Input/Output data of the AIMs belonging the AIW.
Use Case	A particular instance of the Application domain target of an Application Standard.
Version	A revision or extension of a Standard or of one of its elements.

Annex 2 - Notices and Disclaimers Concerning MPAI Standards (Informative)

The notices and legal disclaimers given below shall be borne in mind when downloading and using approved MPAI Standards downloaded from https://www.mpai.community/access/.

In the following, "Standard" means the collection of four documents: "Technical Specification", "Reference Software" and "Conformance Testing" and, where applicable, "Performance Testing" approved and published by MPAI at https://www.mpai.community/resources/.

Life cycle of MPAI Standards

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

MPAI Standards may be modified at any time by corrigenda or new editions. A new edition, however, may not necessarily replace an existing MPAI standard. In order to determine the status of any given MPAI Standard, a user should visit https://mpai.community/access/ web page.

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

Coverage and Applicability of MPAI Standards

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

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

IN NO EVENT SHALL MPAI BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF

THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

MPAI alerts users that practicing its Standards may infringe patents and other rights of third parties.

Users of MPAI Standards should consider all applicable laws and regulations when using an MPAI Standard. The validity of Conformance Testing is strictly technical and refers to the correct implementation of the MPAI Standard. Moreover, positive Performance Assessment of an implementation applies exclusively in the context of the MPAI Governance (https://mpai.community/governance/) and does not imply compliance with any regulatory requirements in the context of any jurisdiction. Therefore, it is the responsibility of the MPAI Standard implementer to observe or refer to the applicable regulatory requirements. By publishing an MPAI Standard, MPAI does not intend to promote actions that are not in compliance with applicable laws, and the Standard shall not be construed as doing so. In particular, users should evaluate MPAI Standards from the viewpoint of data privacy and data ownership in the context of their jurisdictions.

Implementers and users of MPAI Standards documents are responsible for determining and complying with all appropriate safety, security, environmental and health and all applicable laws and regulations.

The Reference Software of an MPAI Standard is released with the MPAI Modified Berkeley Software Distribution licence (link). However, implementers should be aware that the Reference Software of an MPAI Standard may reference some third party software that may have a different licence.

Annex 3 – The Governance of the MPAI Ecosystem (Informative)

1 Level 1 of MPAI standardisation

With reference to *Figure 1*, MPAI issues and maintains a standard – called MPAI-AIF – composed of the following:

1. An environment called AI Framework (AIF) where aggregations of interconnected AI AIWs (AIW) AIMs called AIWs are executed. An AIW implements a use case.
2. AIMs exposing standard interfaces (e.g., access to Controller API) operating as part of an AIW in an AIF.
3. A distribution system of AIFs, AIWs and AIMs called MPAI Store that an AIF can access to download AIWs and AIMs.

MPAI standards include four documents:

1. The *Technical Specification* specifies the elements and operation of the standard and is the main source of information to Implementers.
2. The *Reference Software* is a technically correct implementation of the Technical Specification that can be used as a supplement to the Technical Specification to guide Implementations.
3. The *Conformance Testing* specifies the process, the tools and the data to test the Conformance of an Implementation.
4. The *Performance Assessment* specifies the process, the tools and the data or data specification to test the Performance of the Implementation, i.e., of being Reliable, Robust, Fair and Replicable.

Implementers' Benefits	Upload to the MPAI Store Implementations of MPAI-AIF Upload Implementations of AIWs and AIMs performing proprietary functions. Have a global distribution channel of AIM and AIW implementations that can execute in an AIF Framework.
MPAI Store's role	Tests Implementations for conformance to the MPAI-AIF Technical Specification and verifies the Implementations' security, e.g., absence of malware.

2 Level 2 of MPAI standardisation

MPAI normatively specifies the following aspects of an AI Module (AIM):

4. The format and semantics of the input data, e.g., "video of a talking human face".
5. The function, e.g., "identification of the emotion on the face of and the meaning of the sentence uttered by a speaking human".
6. The format of the output data, e.g., "emotion" and "meaning".

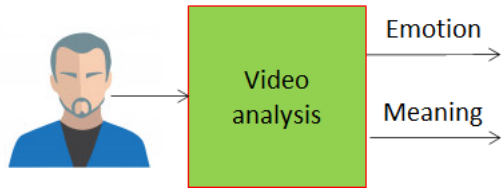

Figure 10 – An AIM with embedded knowledge

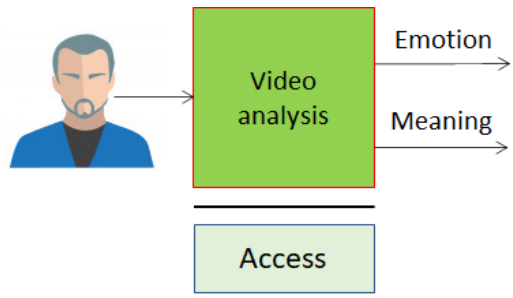

Figure 11 – An AIM with access to an external knowledge base

AIMs can be trained with real data, i.e., made to learn from real data to execute a specific function on new data in the same or similar context (*Figure 10*). The same function, however, can be achieved with an AIM implemented, e.g., with Data Processing technologies (*Figure 11*). If an AIM needs Access to an external knowledge base, MPAI also specifies how the AIM interfaces with it.

MPAI only specifies the mentioned input and output data, and the function of an AIM, but is silent on how the input data are processed.

MPAI Standards are generally agnostic of the implementation technology adopted: hardware, software or hybrid hardware and software.

MPAI Application Standards (as opposed to MPAI-AIF) normatively specify AIWs supporting MPAI-identified Use Cases. An example of a Use Case is "interpreting a sentence from a specified language to another preserving the features of the original sentence", as in *Figure 2* that describes an example of AIW standardised by MPAI.

An MPAI Application Standard specifies:

1. The format and semantics of the input data, e.g., "source speech", "source text" and "desired target language".
2. The function, e.g., "translating a sentence from a language to another, and pronounce it preserving the features of the original speech".
3. The format of the output data, e.g., "speech" and "text".

In a Level 1 implementation of the AIW of *Figure 2* having the depicted input and output data, an implementer can use proprietary AIMs within the constraints of the MPAI-AIF Standard. In a Level 2 implementation, however, the AIW must be implemented with AIMs that conform with an MPAI application standard.

Implementers' benefits	Upload to the MPAI Store Implementations of AIWs and AIMs. Have a global distribution channel of their AIM and AIW Implementations.
Users' benefits	Rely on Implementations having Use Case and AIMs function interfaces that have been reviewed during standardisation. Achieve a level of explainability of AIW operation because the AIM functions and interfaces are known.
Market's benefits	Open AIM markets foster competition leading to better products. Competition of AIM Implementations fosters AI innovation.
MPAI Store's role	Tests Implementations for Conformance with the relevant MPAI Application Standard, and verifies the Implementations' security. Indicates unambiguously that the AIM and AIW Implementations are Level 2

3 Level 3 of MPAI standardisation

MPAI does not generally set standards on how and with what data an AIM should be trained because this is an important differentiator that promotes competition leading to better AI systems. However, the performance of an AIM is typically higher if the data used for training are in greater quantity and more in tune with the scope. Training data that have large variety and cover the spectrum of all cases of interest in breadth and depth typically lead to implementations of higher "quality".

For Level 3, MPAI normatively specifies the process, the tools and the data or the characteristics of the data to be used to Assess the Grade of Performance of an AIM or an AIW.

The definition of Performance is specific to an application domain and is defined in the context of that domain. Unlike Conformance Testing of an Implementation whose outcome has a binary value, Performance Assessment of an Implementation does not necessarily have a binary value. The Performance Assessment of an MPAI standard specifies which of the 4 attributes should be assessed.

Implementers' benefits	Can claim their Implementation has passed Performance Assessment.
Users' benefits	Get assurance the Implementation being used performs correctly, e.g., it has been properly trained.
Market's benefits	Implementation Grades stimulate the development of more Performing AIM and AIW Implementations.
MPAI Store's role	Verifies the Implementations' security. Indicates unambiguously AIM and AIW Implementations are Level 3.

4 The MPAI ecosystem

Figure 12 is a high-level description of the MPAI ecosystem operation applicable to fully conforming MPAI implementations:

1. MPAI establishes and controls the not-for-profit MPAI Store (step 1).
2. MPAI appoints Performance Assessors (step 2).
3. MPAI publishes standards (step 3).
4. Implementer submits an Implementation to a Performance Assessor (step 4).
5. If the Performance of the Implementation is acceptable, Performance Assessor informs the Implementer (step 5a) and the MPAI Store (step 5b).
6. Implementer submits Implementation to the MPAI Store (step 6). Then the Store internally Tests the Conformance and security of the Implementation.
7. User downloads Implementation (step 7).

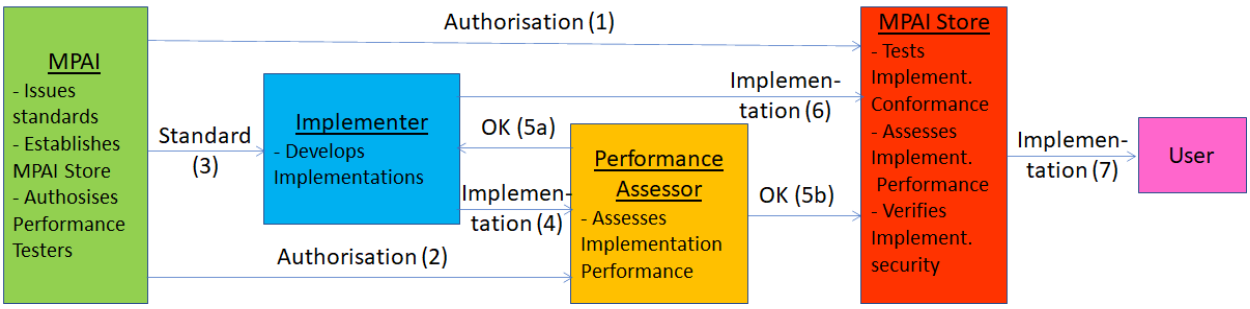

Figure 12 – The MPAI ecosystem operation

The MPAI ecosystem allows an implementer of a standard AIM with outstanding features to access the entire market because users will download the AIM for use in their AIFs. Users will benefit from AIMs produced by a competitive AIM market that builds on and promotes innovation.

Annex 4 – AIW and AIM Metadata of MMC-CWE

1 ID linearization

Note: Fields that are used to generate automatic IDs may not contain "**:**" characters.
When one needs to reference them from other contexts, automatic unique IDs for AIWs/AIMs can be generated with the following formula:

```
AIM->Implementer_ID:(S:AIM->Standard->Name:AIM->Standard->Version:AIM->Standard->Use_Case:AIM->Standard->Name|U:AIM->User_Defined->Name):AIM->Version
```

Examples:

- `00089:(S:(MMC:CWE:2:_MAIN_)):123 // A workflow`
- `00089:(S:(MMC:CWE:2:GovernanceAssessment)):345 // A sub-module of a workflow`
- `00089:(U:PCA_based_analysis):75`

2 AIW metadata for CWE

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store, e.g., 00089
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "1",
                "Name": "_MAIN_" // Always _MAIN_ for workflows
        } | "User_Defined": { // Provided by implementer
                "Name": "Conversation with Emotion"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main" // Defined by MPAI, selected by implementer
        "Description": "This AIW implements CWE application of MPAI-MMC",
        "Ports": [
                {
                        "Name": "Text_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Viddeo_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_4",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_3",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Video_2",
                        "Direction": "OutputInput",
```

```
                "Record_Type": "byte[] bitstream_t",
                "Type": "Software",
                "Protocol": ""
        },
]
"AIMs": [
        // Automatic ID for module 1
        "SpeechRecogniton": "@00089:(S:(MMC:CWE:2:SpeechRecogniton)):345"
        // Automatic ID for module 2
        "VideoAnalysis": "@00089:(S:(MMC:CWE:2:VideoAnalysis)):12",
        // Automatic ID for module 3
        "LanguageUnderstanding": "@00089:(S:(MMC:CWE:2:LanguageUnderstanding)):333",
        // Automatic ID for module 4
        "DialogProcessing": "@00089:(S:(MMC:CWE:2:DialogProcessing)):2",
        // Automatic ID for module 5
        "SpeechSynthesis": "@00089:(S:(MMC:CWE:2:SpeechSynthesis)):27"
        // Automatic ID for module 6
        "LipAnimation": "@00089:(S:(MMC:CWE:2:LipAnimation)):32"
],
"Topology": [
        "Text_1": {
                "Output": {
                        "Module": "",
                        "Port": "Text_1"
                },
                "Input": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Text_1"
                }
        },
        "Text_2": {
                "Output": {
                        "Module": "SpeechRecognition",
                        "Port": "Text_2"
                },
                "Input": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Text_2"
                }
        },
        "Text_3": {
                "Output": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Text_3"
                },
                "Input": {
                        "Module": "DialogProcessing",
                        "Port": "Text_3"
                }
        },
        "Text_4": {
                "Output": {
                        "Module": "DialogProcessing",
                        "Port": "Text_4"
                },
                "Input": {
                        "Module": "",
                        "Port": "Text_4"
                }
        },
        "Speech_1": {
                "Output": {
                        "Module": "",
                        "Port": " Speech_1"
                },
                "Input": {
                        "Module": "SpeechRecognition",
                        "Port": " Speech_1"
                }
        },
        "Speech_2": {
                "Output": {
                        "Module": "SpeechSynthesis",
                        "Port": " Speech_2"
```

```
        },
        "Input": {
                "Module": "LipAnimation",
                "Port": " Speech_2"
        }
},
"Speech_3": {
        "Output": {
                "Module": "SpeechSynthesis",
                "Port": " Speech_3"
        },
        "Input": {
                "Module": "",
                "Port": " Speech_3"
        }
},
"Video_1": {
        "Output": {
                "Module": "",
                "Port": "Video_1"
        },
        "Input": {
                "Module": "VideoAnalysis",
                "Port": "Video_1"
        }
},
"Video_2": {
        "Output": {
                "Module": "LipAnimation",
                "Port": "Video_2"
        },
        "Input": {
                "Module": "",
                "Port": "Video_2"
        }
},
"Meaning_1": {
        "Output": {
                "Module": "LanguageUnderstanding",
                "Port": "Meaning_1"
        },
        "Input": {
                "Module": "DialogProcessing",
                "Port": "Meaning_1"
        }
},
"Meaning_2": {
        "Output": {
                "Module": "VideoAnalysis",
                "Port": "Meaning_2"
        },
        "Input": {
                "Module": "DialogProcessing",
                "Port": "Meaning_2"
        }
},
"Emotion_1": {
        "Output": {
                "Module": "LanguageUnderstanding",
                "Port": "Emotion_1"
        },
        "Input": {
                "Module": "EmotionRecognition",
                "Port": "Emotion_1"
        }
},
"Emotion_2": {
        "Output": {
                "Module": "SpeechRecognition",
                "Port": "Emotion_2"
        },
        "Input": {
                "Module": "EmotionRecognition",
                "Port": "Emotion_2"
```

```
                        }
                },
                "Emotion_3": {
                        "Output": {
                                "Module": "VideoAnalysis",
                                "Port": "Emotion_3"
                        },
                        "Input": {
                                "Module": "EmotionRecognition",
                                "Port": "Emotion_3"
                        }
                },
                "FinalEmotion_1": {
                        "Output": {
                                "Module": "EmotionRecognition",
                                "Port": "FinalEmotion_1"
                        },
                        "Input": {
                                "Module": "DialogProcessing",
                                "Port": "FinalEmotion_1"
                        }
                },
                "FinalEmotion_2": {
                        "Output": {
                                "Module": "DialogProcessing ",
                                "Port": "FinalEmotion_2"
                        },
                        "Input": {
                                "Module": "LipAnimation",
                                "Port": "FinalEmotion_2"
                        }
                },
                "TextWithEmotion": {
                        "Output": {
                                "Module": "DialogProcessing ",
                                "Port": "TextWithEmotion"
                        },
                        "Input": {
                                "Module": "SpeechSynthesis",
                                "Port": "TextWithEmotion"
                        }
                },
        ],
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "ConversationWithEmotion",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3 AIM metadata

3.1 SpeechRecognition

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "2",
                "Name": "SpeechRecognition"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSR"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements speech recognition function that converts
speech to text of user utterance.",
        "Ports": [
                {
                        "Name": "Speech",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
                {
                        "Name": "Emotion_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
                        "Protocol": ""
                }

        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechRecognition",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.2 Video Analysis

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "2",
                "Name": "VideoAnalysis"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYVA"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements video analysis.",
        "Ports": [
                {
                        "Name": "Video_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Emotion_3",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
                {
                        "Name": "Meaning_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }

        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "VideoAnalysis",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.3 Language Understanding

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "2",
                "Name": "LanguageUnderstanding "
        } | "User_Defined": { // Provided by implementer
                "Name": "MYLU"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Language Understanding function.",
        "Ports": [
                {
                        "Name": "Text_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Meaning_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_3",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Emotion_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }

        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "LanguageUnderstanding",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
```

```
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.4 Emotion Recognition

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "2",
                "Name": "EmotionRecognition"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYER"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Emotion Recognition function.",
        "Ports": [
                {
                        "Name": "Emotion_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Emotion_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Emotion_3",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "FinalEmotion_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "EmotionRecognition",
```

```
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.5 Dialog Processing

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "2",
                "Name": "DialogProcessing"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYDP"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Dialog Processing function.",
        "Ports": [
                {
                        "Name": "Meaning_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Meaning_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_3",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "FinalEmotion_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "TextWithEmotion",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "FinalEmotion_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
```

```
                }

        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "DialogProcessing",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.6 Speech Synthesis

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "2",
                "Name": "SpeechSynthesis"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSS"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Synthesis function.",
        "Ports": [
                {
                        "Name": "TextWithEmotion",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_3",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
```

```
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechSynthesis",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.7 Lip Animation

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "CWE",
                "Version": "2",
                "Name": "LipAnimation"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYLA"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Lip Animation function.",
        "Ports": [
                {
                        "Name": "Speech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "FinalEmotion_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Video_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
```

```
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "LipAnimation",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

Annex 5 – AIW and AIM Metadata of MMC-MQA

1 ID linearization

Note: Fields that are used to generate automatic IDs may not contain "**:**" characters.
When one needs to reference them from other contexts, automatic unique IDs for AIWs/AIMs can be generated with the following formula:

```
AIM->Implementer_ID:(S:AIM->Standard->Name:AIM->Standard->Version:AIM->Standard->Use_Case:AIM->Standard->Name|U:AIM->User_Defined->Name):AIM->Version
```

Examples:

- `00089:(S:(MMC:CWE:2:_MAIN_)):123 // A workflow`
- `00089:(S:(MMC:CWE:2:GovernanceAssessment)):345 // A sub-module of a workflow`
- `00089:(U:PCA_based_analysis):75`

2 AIW metadata for MQA

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store, e.g., 00089
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "MQA",
                "Version": "1",
                "Name": "_MAIN_" // Always _MAIN_ for workflows
        } | "User_Defined": { // Provided by implementer
                "Name": "Multimodal Question Answering"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main" // Defined by MPAI, selected by implementer
        "Description": "This AIW implements MQA application of MPAI-MMC",
        "Ports": [
                {
                        "Name": "Text_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Video",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_6",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_2",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                }

        ]
        "AIMs": [
```

```
        // Automatic ID for module 1
        "SpeechRecogniton": "@00089:(S:(MMC:MQA:2:SpeechRecogniton)):345"
        // Automatic ID for module 2
        "VideoAnalysis": "@00089:(S:(MMC:MQA:2:VideoAnalysis)):12",
        // Automatic ID for module 3
        "LanguageUnderstanding": "@00089:(S:(MMC:MQA:2:LanguageUnderstanding)):333",
        // Automatic ID for module 4
        "QuestionAnalysis": "@00089:(S:(MMC:MQA:2:QuestionAnalysis)):2",
       // Automatic ID for module 5
        "QuestionAnswering": "@00089:(S:(MMC:MQA:2:QuestionAnswering)):22",
        // Automatic ID for module 6
        "SpeechSynthesis": "@00089:(S:(MMC:MQA:2:SpeechSynthesis)):27"
],
"Topology": [
        "Text_1": {
                "Output": {
                        "Module": "",
                        "Port": "Text_1"
                },
                "Input": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Text_1"
                }
        },
        "Text_2": {
                "Output": {
                        "Module": "SpeechRecognition",
                        "Port": "Text_2"
                },
                "Input": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Text_2"
                }
        },
        "Text_3": {
                "Output": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Text_3"
                },
                "Input": {
                        "Module": "QuestionAnswering",
                        "Port": "Text_3"
                }
        },
        "Text_4": {
                "Output": {
                        "Module": "SpeechRecognition",
                        "Port": "Text_4"
                },
                "Input": {
                        "Module": "QuestionAnswering",
                        "Port": "Text_4"
                }
        },
        "Text_5": {
                "Output": {
                        "Module": "QuestionAnswering",
                        "Port": "Text_5"
                },
                "Input": {
                        "Module": "SpeechSynthesis",
                        "Port": "Text_5"
                }
        },

        "Text_6": {
                "Output": {
                        "Module": "QuestionAnswering",
                        "Port": "Text_6"
                },
                "Input": {
                        "Module": "",
                        "Port": "Text_6"
                }
```

```
        },

        "Speech_1": {
                "Output": {
                        "Module": "",
                        "Port": " Speech_1"
                },
                "Input": {
                        "Module": "SpeechRecognition",
                        "Port": " Speech_1"
                }
        },
        "Speech_2": {
                "Output": {
                        "Module": "SpeechSynthesis",
                        "Port": " Speech_2"
                },
                "Input": {
                        "Module": "",
                        "Port": " Speech_2"
                }
        },
        "Video": {
                "Output": {
                        "Module": "",
                        "Port": "Video"
                },
                "Input": {
                        "Module": "VideoAnalysis",
                        "Port": "Video"
                }
        },
        "Meaning_1": {
                "Output": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Meaning_1"
                },
                "Input": {
                        "Module": "QuestionAnswering",
                        "Port": "Meaning_1"
                }
        },
        "Meaning_2": {
                "Output": {
                        "Module": "LanguageUnderstanding",
                        "Port": "Meaning_2"
                },
                "Input": {
                        "Module": "QuestionAnalysis",
                        "Port": "Meaning_2"
                }
        },
        "Intention": {
                "Output": {
                        "Module": " QuestionAnalysis",
                        "Port": "Intention"
                },
                "Input": {
                        "Module": "QuestionAnswering",
                        "Port": "Intention"
                }
        },
        "ObjectIdentifier": {
                "Output": {
                        "Module": "VideoAnalysis",
                        "Port": "ObjectIdentifier"
                },
                "Input": {
                        "Module": "EmotionRecognition",
                        "Port": "ObjectIdentifier"
                }
        }
],
"Authentication": "ENC.V.?",
```

```
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "MultimodalQuestionAnswering",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3 AIM metadata

3.1 SpeechRecognition

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "MQA",
                "Version": "2",
                "Name": "SpeechRecognition"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSR2"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements speech recognition function that converts
speech to text of user utterance.",
        "Ports": [
                {
                        "Name": "Speech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
                {
                        "Name": "Text_4",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
                        "Protocol": ""
                }

        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
```

```
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechRecognition",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.2 Video Analysis

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "MQA",
                "Version": "2",
                "Name": "VideoAnalysis"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYVA2"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements video analysis.",
        "Ports": [
                {
                        "Name": "Video",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "ObjectIdentifier",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
```

```
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "VideoAnalysis",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.3 Language Understanding

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "MQA",
                "Version": "2",
                "Name": "LanguageUnderstanding "
        } | "User_Defined": { // Provided by implementer
                "Name": "MYLU2"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Language Understanding function for MQA.",
        "Ports": [
                {
                        "Name": "Text_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "ObjectIdentifier",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Meaning_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Meaning_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_3",
```

```
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "LanguageUnderstanding",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.4 Question Analysis

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "MQA",
                "Version": "2",
                "Name": "QuestionAnalysis"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYQA"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Question Analysis function.",
        "Ports": [
                {
                        "Name": "Meaning_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Intention",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
```

```
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "QuestionAnalysis",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.5 Question Answering

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "MQA",
                "Version": "2",
                "Name": "QuestionAnswering"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYQANS"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Question Answering function.",
        "Ports": [
                {
                        "Name": "Meaning_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_3",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_4",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_5",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
```

```
                        "Name": "Text_6",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "QuestionAnswering",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.6 Speech Synthesis (Text)

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "MQA",
                "Version": "2",
                "Name": "SpeechSynthesis2"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSS"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Synthesis function for MQA.",
        "Ports": [
                {
                        "Name": "Text_5",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
```

```
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechSynthesis2",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

Annex 6 – AIW and AIM Metadata of MMC-UST

1 ID linearization

Note: Fields that are used to generate automatic IDs may not contain "**:**" characters.
When one needs to reference them from other contexts, automatic unique IDs for AIWs/AIMs can be generated with the following formula:

```
AIM->Implementer_ID:(S:AIM->Standard->Name:AIM->Standard->Version:AIM->Standard->Use_Case:AIM->Standard->Name|U:AIM->User_Defined->Name):AIM->Version
```

Examples:

- `00089:(S:(MMC:CWE:2:_MAIN_)):123 // A workflow`
- `00089:(S:(MMC:CWE:2:GovernanceAssessment)):345 // A sub-module of a workflow`
- `00089:(U:PCA_based_analysis):75`

2 AIW metadata for UST

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store, e.g., 00089
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "UST",
                "Version": "1",
                "Name": "_MAIN_" // Always _MAIN_ for workflows
        } | "User_Defined": { // Provided by implementer
                "Name": "Unidirectional Speech Translation"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main" // Defined by MPAI, selected by implementer
        "Description": "This AIW implements UST application of MPAI-MMC",
        "Ports": [
                {
                        "Name": "RequestedLanguage",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceText",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_3",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech",
                        "Direction": "OutputInput",
```

```
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                }
        ]
        "AIMs": [
                // Automatic ID for module 1
                "SpeechRecogniton": "@00089:(S:(MMC:UST:2:SpeechRecogniton)):345"
                // Automatic ID for module 2
                "Translation": "@00089:(S:(MMC:UST:2:Translation)):12",
                // Automatic ID for module 3
                "SpeechFeatureExtraction":
"@00089:(S:(MMC:UST:2:LanguageUnderstanding)):333",
                // Automatic ID for module 4
                “SpeechSynthesis": "@00089:(S:(MMC:UST:2:SpeechSynthesis)):27"
                ],
        "Topology": [
                "RequestedLanguage": {
                        "Output": {
                                "Module": "",
                                "Port”: "RequestedLanguage"
                        },
                        "Input": {
                                "Module": "Translation",
                                "Port": "RequestedLanguage"
                        }
                },
                "SourceText": {
                        "Output": {
                                "Module": "",
                                "Port”: "SourceText"
                        },
                        "Input": {
                                "Module": "Translation",
                                "Port": "SourceText "
                        }
                },
                "SourceSpeech_1": {
                        "Output": {
                                "Module": "",
                                "Port”: “SourceSpeech_1"
                        },
                        "Input": {
                                "Module": "SpeechRecognition",
                                "Port": “SourceSpeech_1"
                        }
                },
                "SourceSpeech_2": {
                        "Output": {
                                "Module": "",
                                "Port”: “SourceSpeech_2"
                        },
                        "Input": {
                                "Module": "SpeechFeatureExtraction",
                                "Port": “SourceSpeech_2"
                        }
                },
                "Speech": {
                        "Output": {
                                "Module": "SpeechSynthesis",
                                "Port”: " Speech"
                        },
                        "Input": {
                                "Module": "",
                                "Port": "Speech"
                        }
                },
                "SpeechFeatures": {
                        "Output": {
                                "Module": "SpeechFeatureExtraction",
                                "Port”: "SpeechFeatures"
                        },
                        "Input": {
                                "Module": "SpeechSynthesis",
```

```
                        "Port": "SpeechFeatures"
                }
        },

        "Text_1": {
                "Output": {
                        "Module": "SpeechRecognition",
                        "Port": "Text_1"
                },
                "Input": {
                        "Module": "Translation",
                        "Port": "Text_1"
                }
        },
        "Text_2": {
                "Output": {
                        "Module": "Translation",
                        "Port": "Text_2"
                },
                "Input": {
                        "Module": "SpeechSynthesis",
                        "Port": "Text_2"
                }
        }
        "Text_3": {
                "Output": {
                        "Module": "Translation",
                        "Port": "Text_3"
                },
                "Input": {
                        "Module": "",
                        "Port": "Text_3"
                }
        }
    ],
    "Authentication": "ENC.V.?",
    "TimeBase": "PROT.V.?",
    "ResourcePolicies": [
        "CPU": ?
    ],
    "UserAPIProfile": "Low.V",
    "ControllerAPIProfile": {
        Version: "27",
        Level: "High"
    },
    "Implementations": [
        {
            "Type": "Source",
            "Function_Name": "UnidirectionalSpeechTranslation",
            "Language": "C",
            "Architecture": "",
            "OS": "",
            "OS_Version": "",
            "ID": ""
        }
    ],
    "Documentation": [
        { "Type": "tutorial",
          "URI": https://mpai.community/standards/mpai-mmc/
        }
    ]
}
```

3 AIM metadata

3.1 SpeechRecognition

```
"AIM": {
    "Implementer_ID": ###, // Number provided by MPAI store
    "Standard": { // Defined by MPAI, selected by implementer
        "Name": "MMC",
        "Use_Case": "UST",
        "Version": "2",
```

```
                "Name": "SpeechRecognition3"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSR3"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements speech recognition function for UST that
converts speech to text of user utterance.",
        "Ports": [
                {
                        "Name": "SourceSpeech",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechRecognition3",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.2 Translation

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "UST",
                "Version": "2",
                "Name": "Translation"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYTR"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Translation function.",
        "Ports": [
```

```
                {
                        "Name": "RequestedLanguage",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceText",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
                {
                        "Name": "Text_3",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }

        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "Translation",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.3 Speech Feature Extraction

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "UST",
                "Version": "2",
                "Name": "SpeechFeatureExtraction "
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSFE"
```

```
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Feature Extraction function.",
        "Ports": [
                {
                        "Name": "SourceSpeech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechFeatureExtraction",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.4 Speech Synthesis

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "UST",
                "Version": "2",
                "Name": "SpeechSynthesis"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSS"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Synthesis function.",
        "Ports": [
                {
                        "Name": "Text_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
```

```
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechSynthesis",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

Annex 7 – AIW and AIM Metadata of MMC-BST

1 ID linearization

Note: Fields that are used to generate automatic IDs may not contain "**:**" characters.
When one needs to reference them from other contexts, automatic unique IDs for AIWs/AIMs can be generated with the following formula:

```
AIM->Implementer_ID:(S:AIM->Standard->Name:AIM->Standard->Version:AIM->Standard->Use_Case:AIM->Standard->Name|U:AIM->User_Defined->Name):AIM->Version
```

Examples:

- `00089:(S:(MMC:CWE:2:_MAIN_)):123 // A workflow`
- `00089:(S:(MMC:CWE:2:GovernanceAssessment)):345 // A sub-module of a workflow`
- `00089:(U:PCA_based_analysis):75`

2 AIW metadata for BST

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store, e.g., 00089
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "BST",
                "Version": "1",
                "Name": "_MAIN_" // Always _MAIN_ for workflows
        } | "User_Defined": { // Provided by implementer
                "Name": "Bidirectional Speech Translation"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main" // Defined by MPAI, selected by implementer
        "Description": "This AIW implements BST application of MPAI-MMC",
        "Ports": [
                {
                        "Name": "RequestedLanguage",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceText_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceText_2",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
```

```
                {
                        "Name": "SourceSpeech_3",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_4",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },

                {
                        "Name": "TranslationResult_3",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "TranslationResult_4",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Translated_Speech_1",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Translated_Speech_2",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                }

        ]
        "AIMs": [
                // Automatic ID for module 1
                "SpeechRecogniton": "@00089:(S:(MMC:BST:2:SpeechRecogniton)):345"
                // Automatic ID for module 2
                "Translation": "@00089:(S:(MMC:BST:2:Translation)):12",
                // Automatic ID for module 3
                "SpeechFeatureExtraction":
"@00089:(S:(MMC:BST:2:LanguageUnderstanding)):333",
                // Automatic ID for module 4
                “SpeechSynthesis": "@00089:(S:(MMC:BST:2:SpeechSynthesis)):27"
                ],
        "Topology": [
                " RequestedLanguage": {
                        "Output": {
                                "Module": "",
                                "Port”: " RequestedLanguage "
                        },
                        "Input": {
                                "Module": "Translation",
                                "Port": " RequestedLanguage"
                        }
                },
                "SourceText_1": {
                        "Output": {
                                "Module": "",
                                "Port”: "SourceText_1"
                        },
                        "Input": {
                                "Module": "Translation",
```

```
                "Port": "SourceText_1"
        }
},
"SourceText_2": {
        "Output": {
                "Module": "",
                "Port": "SourceText_2"
        },
        "Input": {
                "Module": "Translation",
                "Port": "SourceText_2"
        }
},
"SourceSpeech_1": {
        "Output": {
                "Module": "",
                "Port": “SourceSpeech_1"
        },
        "Input": {
                "Module": "SpeechRecognition",
                "Port": “SourceSpeech_1"
        }
},
"SourceSpeech_2": {
        "Output": {
                "Module": "",
                "Port": “SourceSpeech_2"
        },
        "Input": {
                "Module": "SpeechRecognition",
                "Port": “SourceSpeech_2"
        }
},
"SourceSpeech_3": {
        "Output": {
                "Module": "",
                "Port": “SourceSpeech_3"
        },
        "Input": {
                "Module": "SpeechFeatureExtraction",
                "Port": “SourceSpeech_3"
        }
},
"SourceSpeech_4": {
        "Output": {
                "Module": "",
                "Port": “SourceSpeech_4"
        },
        "Input": {
                "Module": "SpeechFeatureExtraction",
                "Port": “SourceSpeech_4"
        }
},
"TranslatedSpeech_1": {
        "Output": {
                "Module": "SpeechSynthesis",
                "Port": "TranslatedSpeech_1"
        },
        "Input": {
                "Module": "",
                "Port": "TranslatedSpeech_1"
        }
},
"TranslatedSpeech_2": {
        "Output": {
                "Module": "SpeechSynthesis",
                "Port": "TranslatedSpeech_2"
        },
        "Input": {
                "Module": "",
                "Port": "TranslatedSpeech_2"
        }
},
"SpeechFeatures_1": {
```

```
        "Output": {
                "Module": "SpeechFeatureExtraction",
                "Port": " SpeechFeatures_1"
        },
        "Input": {
                "Module": "SpeechSynthesis",
                "Port": "SpeechFeatures_1"
        }
},
"SpeechFeatures_2": {
        "Output": {
                "Module": "SpeechFeatureExtraction",
                "Port": "SpeechFeatures_2"
        },
        "Input": {
                "Module": "SpeechSynthesis",
                "Port": "SpeechFeatures_2"
        }
},
"Text_1": {
        "Output": {
                "Module": "SpeechRecognition",
                "Port": "Text_1"
        },
        "Input": {
                "Module": "Translation",
                "Port": "Text_1"
        }
},
"Text_2": {
        "Output": {
                "Module": "SpeechRecognition",
                "Port": "Text_2"
        },
        "Input": {
                "Module": "Translation",
                "Port": "Text_2"
        }
},
"TranslationResult_1": {
        "Output": {
                "Module": "Translation",
                "Port": "TranslationResult_1"
        },
        "Input": {
                "Module": "SpeechSynthesis",
                "Port": "TranslationResult_1"
        }
},
"TranslationResult_2": {
        "Output": {
                "Module": "Translation",
                "Port": "TranslationResult_2"
        },
        "Input": {
                "Module": "SpeechSynthesis",
                "Port": "TranslationResult_2"
        }
},
"TranslationResult_3": {
        "Output": {
                "Module": "Translation",
                "Port": "TranslationResult_3"
        },
        "Input": {
                "Module": "",
                "Port": "TranslationResult_3"
        }
},,
"TranslationResult_4": {
        "Output": {
                "Module": "Translation",
                "Port": "TranslationResult_4"
        },
```

```
                        "Input": {
                                "Module": "",
                                "Port": "TranslationResult_4"
                        }
                }
        ],
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "BidirectionalSpeechTranslation",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3 AIM metadata

3.1 SpeechRecognition

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "BST",
                "Version": "2",
                "Name": "SpeechRecognition3"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSR4"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements speech recognition function for BST that
converts speech to text of user utterance.",
        "Ports": [
                {
                        "Name": "SourceSpeech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
```

```
                        "Protocol": ""
                }

                {
                        "Name": "Text_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechRecognition4",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.2 Translation

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "UST",
                "Version": "2",
                "Name": "Translation"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYTR"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Translation function.",
        "Ports": [
                {
                        "Name": "RequestedLanguage",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceText_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
```

```
        },
        {
                "Name": "SourceText_2",
                "Direction": "InputOutput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "Text_1",
                "Direction": "InputOutput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "Text_2",
                "Direction": "InputOutput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "TranslationResult_1",
                "Direction": "OutputInput",
                "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "TranslationResult_2",
                "Direction": "OutputInput",
                "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "TranslationResult_3",
                "Direction": "OutputInput",
                "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "TranslationResult_4",
                "Direction": "OutputInput",
                "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                "Type": "Software",
                "Protocol": ""
        }
],
"AIMs": [ ], // Does not depend on other AIMs
"Topology": [ ], // Does not depend on other AIMs
"Authentication": "ENC.V.?",
"TimeBase": "PROT.V.?",
"ResourcePolicies": [
        "CPU": ?
],
"UserAPIProfile": "Low.V",
"ControllerAPIProfile": {
        Version: "27",
        Level: "High"
},
"Implementations": [
        {
                "Type": "Source",
                "Function_Name": "Translation",
                "Language": "C",
                "Architecture": "",
                "OS": "",
                "OS_Version": "",
                "ID": ""
        }
],
```

```
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.3 Speech Feature Extraction

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "BST",
                "Version": "2",
                "Name": "SpeechFeatureExtraction "
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSFE"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Feature Extraction function.",
        "Ports": [
                {
                        "Name": "SourceSpeech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }

        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechFeatureExtraction",
                        "Language": "C",
                        "Architecture": "",
```

```
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.4 Speech Synthesis

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "UST",
                "Version": "2",
                "Name": "SpeechSynthesis"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSS"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Synthesis function.",
        "Ports": [
                {
                        "Name": "TranslationResult_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "TranslationResult_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "TranslatedSpeech_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "TranslatedSpeech_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
```

```
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechSynthesis",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

Annex 8 – AIW and AIM Metadata of MMC-OMT

1 ID linearization

Note: Fields that are used to generate automatic IDs may not contain "**:**" characters.
When one needs to reference them from other contexts, automatic unique IDs for AIWs/AIMs can be generated with the following formula:

```
AIM->Implementer_ID:(S:AIM->Standard->Name:AIM->Standard->Version:AIM->Standard->Use_Case:AIM->Standard->Name|U:AIM->User_Defined->Name):AIM->Version
```

Examples:

- `00089:(S:(MMC:CWE:2:_MAIN_)):123 // A workflow`
- `00089:(S:(MMC:CWE:2:GovernanceAssessment)):345 // A sub-module of a workflow`
- `00089:(U:PCA_based_analysis):75`

2 AIW metadata for OMT

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store, e.g., 00089
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "OMT",
                "Version": "1",
                "Name": "_MAIN_" // Always _MAIN_ for workflows
        } | "User_Defined": { // Provided by implementer
                "Name": "OneToManySpeechTranslation"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main" // Defined by MPAI, selected by implementer
        "Description": "This AIW implements OMT application of MPAI-MMC",
        "Ports": [
                {
                        "Name": "RequestedLanguage",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceText",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_1",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SourceSpeech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_3",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
```

```
                {
                        "Name": "Text_N",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_1",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_2",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_N",
                        "Direction": "OutputInput",
                        "Record_Type": "byte[] bitstream_t",
                        "Type": "Software",
                        "Protocol": ""
                }
        ]
        "AIMs": [
                // Automatic ID for module 1
                "SpeechRecogniton": "@00089:(S:(MMC:OMT:2:SpeechRecogniton)):345"
                // Automatic ID for module 2
                "Translation": "@00089:(S:(MMC:OMT:2:Translation)):12",
                // Automatic ID for module 3
                "SpeechFeatureExtraction":
"@00089:(S:(MMC:UST:2:LanguageUnderstanding)):333",
                // Automatic ID for module 4
                “SpeechSynthesis": "@00089:(S:(MMC:OMT:2:SpeechSynthesis)):27"
                ],
        "Topology": [
                "RequestedLanguage": {
                        "Output": {
                                "Module": "",
                                "Port”: "RequestedLanguage"
                        },
                        "Input": {
                                "Module": "Translation",
                                "Port": "RequestedLanguage"
                        }
                },
                "SourceText": {
                        "Output": {
                                "Module": "",
                                "Port”: "SourceText"
                        },
                        "Input": {
                                "Module": "Translation",
                                "Port": "SourceText "
                        }
                },
                "SourceSpeech_1": {
                        "Output": {
                                "Module": "",
                                "Port”: “SourceSpeech_1"
                        },
                        "Input": {
                                "Module": "SpeechRecognition",
                                "Port": “SourceSpeech_1"
                        }
                },
                "SourceSpeech_2": {
                        "Output": {
```

```
                "Module": "",
                "Port": “SourceSpeech_2"
        },
        "Input": {
                "Module": "SpeechFeatureExtraction",
                "Port": “SourceSpeech_2"
        }
},
"Speech_1": {
        "Output": {
                "Module": "SpeechSynthesis",
                "Port": " Speech_1"
        },
        "Input": {
                "Module": "",
                "Port": "Speech_1"
        }
},
"Speech_2": {
        "Output": {
                "Module": "SpeechSynthesis",
                "Port": " Speech_2"
        },
        "Input": {
                "Module": "",
                "Port": "Speech_2"
        }
},
"Speech_N": {
        "Output": {
                "Module": "SpeechSynthesis",
                "Port": " Speech_N"
        },
        "Input": {
                "Module": "",
                "Port": "Speech_N"
        }
},

"SpeechFeatures": {
        "Output": {
                "Module": "SpeechFeatureExtraction",
                "Port": "SpeechFeatures"
        },
        "Input": {
                "Module": "SpeechSynthesis",
                "Port": "SpeechFeatures"
        }
},

"Text_1": {
        "Output": {
                "Module": "SpeechRecognition",
                "Port": "Text_1"
        },
        "Input": {
                "Module": "Translation",
                "Port": "Text_1"
        }
},
"TranslationResult_1": {
        "Output": {
                "Module": "Translation",
                "Port": "TranslationResult_1"
        },
        "Input": {
                "Module": "SpeechSynthesis",
                "Port": "TranslationResult_1"
        }
}
"TranslationResult_2": {
        "Output": {
                "Module": "Translation",
                "Port": "TranslationResult_2"
```

```
                },
                "Input": {
                        "Module": "SpeechSynthesis",
                        "Port": "TranslationResult_2"
                }
        }

        "TranslationResult_N": {
                "Output": {
                        "Module": "Translation",
                        "Port": "TranslationResult_N"
                },
                "Input": {
                        "Module": "SpeechSynthesis",
                        "Port": "TranslationResult_N"
                }
        }
        "Text_2": {
                "Output": {
                        "Module": "Translation",
                        "Port": "Text_2"
                },
                "Input": {
                        "Module": "",
                        "Port": "Text_2"
                }
        }
        "Text_3": {
                "Output": {
                        "Module": "Translation",
                        "Port": "Text_3"
                },
                "Input": {
                        "Module": "",
                        "Port": "Text_3"
                }
        }
        "Text_N": {
                "Output": {
                        "Module": "Translation",
                        "Port": "Text_N"
                },
                "Input": {
                        "Module": "",
                        "Port": "Text_N"
                }
        }
],
"Authentication": "ENC.V.?",
"TimeBase": "PROT.V.?",
"ResourcePolicies": [
        "CPU": ?
],
"UserAPIProfile": "Low.V",
"ControllerAPIProfile": {
        Version: "27",
        Level: "High"
},
"Implementations": [
        {
                "Type": "Source",
                "Function_Name": "OneToManySpeechTranslation",
                "Language": "C",
                "Architecture": "",
                "OS": "",
                "OS_Version": "",
                "ID": ""
        }
],
"Documentation": [
        { "Type": "tutorial",
          "URI": https://mpai.community/standards/mpai-mmc/
        }
]
```

```
}
```

3 AIM metadata

3.1 SpeechRecognition

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "OMT",
                "Version": "2",
                "Name": "SpeechRecognition5"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSR5"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements speech recognition function for OMT that
converts speech to text of user utterance.",
        "Ports": [
                {
                        "Name": "SourceSpeech",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[]
frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechRecognition5",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.2 Translation

```
"AIM": {
```

```
"Implementer_ID": ###, // Number provided by MPAI store
"Standard": { // Defined by MPAI, selected by implementer
        "Name": "MMC",
        "Use_Case": "OMT",
        "Version": "2",
        "Name": "Translation"
} | "User_Defined": { // Provided by implementer
        "Name": "MYTR"
},
"Version": "345", // Provided by implementer
"Profile": "Main", // Defined by MPAI, selected by implementer
"Description": "This AIM implements Translation function.",
"Ports": [
        {
                "Name": "RequestedLanguage",
                "Direction": "InputOutput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "SourceText",
                "Direction": "InputOutput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "Text_1",
                "Direction": "InputOutput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "TranslationResult_1",
                "Direction": "OutputInput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "TranslationResult_2",
                "Direction": "OutputInput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        },
        {
                "Name": "TranslationResult_N",
                "Direction": "OutputInput",
                "Record_Type": "{int32 modelSize ; byte[] model}",
                "Type": "Software",
                "Protocol": ""
        }

],
"AIMs": [ ], // Does not depend on other AIMs
"Topology": [ ], // Does not depend on other AIMs
"Authentication": "ENC.V.?",
"TimeBase": "PROT.V.?",
"ResourcePolicies": [
        "CPU": ?
],
"UserAPIProfile": "Low.V",
"ControllerAPIProfile": {
        Version: "27",
        Level: "High"
},
"Implementations": [
        {
                "Type": "Source",
                "Function_Name": "Translation",
                "Language": "C",
```

```
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```

3.3 Speech Feature Extraction

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "OMT",
                "Version": "2",
                "Name": "SpeechFeatureExtraction "
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSFE"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Feature Extraction function for OMT.",
        "Ports": [
                {
                        "Name": "SourceSpeech_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 frameNumber; int16 x; int16 y; byte[] frame}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechFeatureExtraction",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
```

```
        ]
}
```

3.4 Speech Synthesis

```
"AIM": {
        "Implementer_ID": ###, // Number provided by MPAI store
        "Standard": { // Defined by MPAI, selected by implementer
                "Name": "MMC",
                "Use_Case": "OMT",
                "Version": "2",
                "Name": "SpeechSynthesis"
        } | "User_Defined": { // Provided by implementer
                "Name": "MYSS"
        },
        "Version": "345", // Provided by implementer
        "Profile": "Main", // Defined by MPAI, selected by implementer
        "Description": "This AIM implements Speech Synthesis function for OMT.",
        "Ports": [
                {
                        "Name": "TranslationResult_1",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "TranslationResult_2",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "TranslationResult_N",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "SpeechFeatures",
                        "Direction": "InputOutput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_1",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Text_N",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_1",
                        "Direction": "OutputInput",
```

```
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_2",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                },
                {
                        "Name": "Speech_N",
                        "Direction": "OutputInput",
                        "Record_Type": "{int32 modelSize ; byte[] model}",
                        "Type": "Software",
                        "Protocol": ""
                }
        ],
        "AIMs": [ ], // Does not depend on other AIMs
        "Topology": [ ], // Does not depend on other AIMs
        "Authentication": "ENC.V.?",
        "TimeBase": "PROT.V.?",
        "ResourcePolicies": [
                "CPU": ?
        ],
        "UserAPIProfile": "Low.V",
        "ControllerAPIProfile": {
                Version: "27",
                Level: "High"
        },
        "Implementations": [
                {
                        "Type": "Source",
                        "Function_Name": "SpeechSynthesis",
                        "Language": "C",
                        "Architecture": "",
                        "OS": "",
                        "OS_Version": "",
                        "ID": ""
                }
        ],
        "Documentation": [
                { "Type": "tutorial",
                  "URI": https://mpai.community/standards/mpai-mmc/
                }
        ]
}
```