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

MPAI Technical Specification

Multimodal Conversation
MPAI-MMC

V1

WARNING

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

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

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

© Copyright MPAI 2021. All rights reserved

Multimodal Conversation

1 Introduction 4
2 Scope of Standard 5
2.1 Conversation With Emotion (CWE) 5
2.2 Multimodal Question Answering (MQA) 5
2.3 Unidirectional Speech Translation (UST) 6
2.4 Bidirectional Speech Translation (BST) 6
2.5 One-to-Many Speech Translation (MST) 6
2.6 Normative content of the Use Cases 6
3 Terms and Definitions 7
4 References 8
4.1 Normative References 8
4.2 Informative References 8
5 Use Case Architectures 8
5.1 Conversation with Emotion (CWE) 8
5.1.1 Scope of Use Case 8
5.1.2 Input/Output Data 9
5.1.3 Implementation Architecture 9
5.1.4 AI Modules 10
5.1.5 AIW Metadata 10
5.2 Multimodal Question Answering (MQA) 10
5.2.1 Scope of standard 10
5.2.2 Input/output data 10
5.2.3 Implementation Architecture 10
5.2.4 AI Modules 11
5.2.5 AIW Metadata 11
5.3 Unidirectional Speech Translation (UST) 11
5.3.1 Scope of Use Case 11
5.3.2 Input/output data 12
5.3.3 Implementation Architecture 12
5.3.4 AI Modules 12
5.3.5 AIW Metadata 13
5.4 Bidirectional Speech Translation (BST) 13
5.4.1 Scope of Use Case 13
5.4.2 Input/output data 13
5.4.3 Implementation Architecture 13
5.4.4 AI Modules 14
5.4.5 AIW Metadata 14
5.5 One-to-Many Speech Translation (MST) 14
5.5.1 Scope of Use Case 14
5.5.2 Input/output data 15
5.5.3 Implementation Architecture 15
5.5.4 AI Modules 15
5.5.5 AIW Metadata 16
6 AI Modules 16
6.1 AI Module Interoperability 16
6.2 MPAI-MMC AIMs and their data 16
6.2.1 Conversation with Emotion (CWE) 16

6.2.2 Multimodal Question Answering (MQA) 17
6.2.3 Unidirectional Speech Translation (UST) 17
6.2.4 Bidirectional Speech Translation (BST) 18
6.2.5 One-to-Many Speech Translation (MST) 18
6.3 Data Formats 18
6.3.1 Emotion 19
6.3.2 Intention 23
6.3.3 Language identifier 25
6.3.4 Meaning 25
6.3.5 Object Identifier 26
6.3.6 Speech 27
6.3.7 Speech Features 27
6.3.8 Text 30
6.3.9 Text with Emotion 30
6.3.10 Video 30
6.3.11 Video File 31
6.3.12 Video of Faces KB Query Format 31
Annex 1 – MPAI-wide terms and definitions (Normative) 32
Annex 2 - Notices and Disclaimers Concerning MPAI Standards (Informative) 35
Annex 3 – The Governance of the MPAI Ecosystem (Informative) 37
Annex 4 – AIW and AIM Metadata of MMC-CWE 39
1 AIW metadata for CWE 39
2 AIM metadata 43
2.1 SpeechRecognition 43
2.2 Video Analysis 43
2.3 Language Understanding 44
2.4 Emotion Fusion 45
2.5 Dialog Processing 46
2.6 Speech Synthesis 47
2.7 Lips Animation 48
Annex 5 – AIW and AIM Metadata of MMC-MQA 50
1 AIW metadata for MQA 50
2 AIM metadata 53
2.1 SpeechRecognition 53
2.2 Video Analysis 54
2.3 Language Understanding 55
2.4 Question Analysis 56
2.5 Question Answering 56
2.6 Speech Synthesis (Text) 57
Annex 6 – AIW and AIM Metadata of MMC-UST 59
1 AIW metadata for UST 59
2 AIM metadata 61
2.1 SpeechRecognition 61
2.2 Translation 62
2.3 Speech Feature Extraction 63
2.4 Speech Synthesis 64
Annex 7 – AIW and AIM Metadata of MMC-BST 65
1 AIW metadata for BST 65
2 AIM metadata 69
2.1 SpeechRecognition 69

2.2 Translation 70
2.3 Speech Feature Extraction 71
2.4 Speech Synthesis 72
Annex 8 – AIW and AIM Metadata of MMC-MST 74
1 AIW metadata for MST 74
2 AIM metadata 77
2.1 SpeechRecognition 77
2.2 Translation 78
2.3 Speech Feature Extraction 79
2.4 Speech Synthesis 80
Annex 9 – Communication Among AIM Implementors 82

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. MPAI Application Standards enable the development of AI-based products, applications and services.

In the following, Terms beginning with a capital letter are defined in *Table 1* if they are specific to this Standard and in *Table 21* if they are common to all MPAI Standards.

MPAI Application Standards Implementations operate in the AI Framework (AIF) specified by the MPAI-AIF Standard (MPAI-AIF). *Figure 1* is the MPAI-AIF Reference Model.

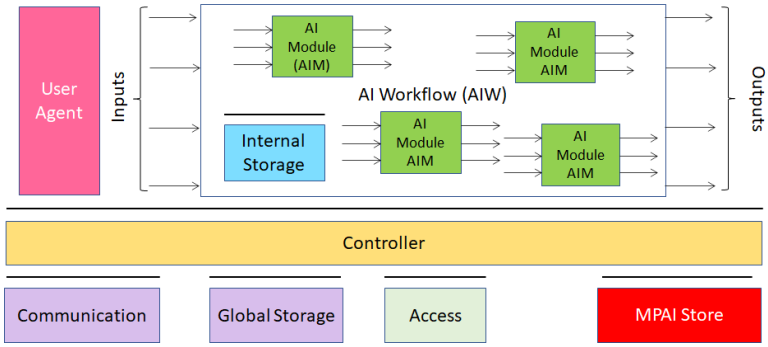

Figure 1 – The AI Framework (AIF) Reference Model and its Components

An AIF Implementation allows execution of AI Workflows (AIW), composed by basic processing elements called AI Modules (AIM).

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

In particular, an AIM is defined by its Function and Data, but not by its internal architecture, which may be based on AI or data processing, and implemented in software, hardware or hybrid software and hardware technologies.

MPAI defines Interoperability as the ability to replace an AIF, an AIW or an AIM Implementation with a functionally equivalent Implementation. MPAI also defines 3 Interoperability Levels of an AIF that executes an AIW. The AIW may be:

1. Proprietary and composed of AIMs with proprietary functions using any proprietary data Format (*Level 1*).

2. Composed of AIMs having all their Functions, Formats and Connections specified by an MPAI Application Standard (*Level 2*).
3. Composed of AIMs that have the characteristics of point 2. above and certified by an MPAI-appointed Assessor to possess the attributes of Reliability, Robustness, Replicability and Fairness – collectively called Performance (*Level 3*).

MPAI is the root of trust of the MPAI Ecosystem [1] offering Users access to the promised benefits of AI with a guarantee of increased transparency, trust and reliability as the Interoperability Level of an Implementation moves from 1 to 3. Additional information is provided by Annex 3.

2 Scope of Standard

Conversation with Emotion (MPAI-MMC) is an MPAI Standard comprising five Use Cases, all sharing the use of AI to enable a form of human-machine conversation that emulates human-human conversation in completeness and intensity:

1. "Conversation with Emotion" (CWE), supporting audio-visual conversation with a machine impersonated by a synthetic voice and an animated face.
2. "Multimodal Question Answering" (MQA), supporting request for information about a displayed object.
3. Three Uses Cases support conversational translation applications. In each Use Case, users can specify whether speech or text is used as input and, if it is speech, whether their speech features are preserved in the interpreted speech:
    a. "Unidirectional Speech Translation" (UST).
    b. "Bidirectional Speech Translation" (BST).
    c. "One-to-Many Speech Translation" (MST).

The current Version of MPAI-MMC has been developed by the MPAI Multimodal Conversation Development Committee (MM-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.

MPAI expects to produce future MPAI-MMC Versions supporting enhanced current and new Use Cases.

2.1 Conversation With Emotion (CWE)

When people talk, they use multiple modalities. Emotion is a key feature in understanding the meaning of speakers' utterances. Therefore, a conversation system capable of recognising and conveying emotion can better foster understanding of the user and produce better replies.

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

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 cases where input is in the form of text or speech. However, there can also be cases where mixed inputs are presented to the system, for instance including both speech and images. For example, a user might ask "Where can I buy this tool?" while showing the picture of the tool. In the Multimodal Question Answering (MMC-MQA) Use Case, a machine responds to a question expressed by a user in text or speech while showing an object. The machine's response may use text and synthetic speech.

2.3 Unidirectional Speech Translation (UST)

In the Unidirectional Speech Translation (MMC-UST) Use Case, the system recognises the text of a speech segment uttered in a specified language; passes that recognised text through automatic translation into another specified language; and outputs a spoken version of the translated text as subtitles and/or as a synthesised voice. If the desired output is speech, users can specify whether their speech features (voice colour, emotional charge, etc.) are preserved in the interpreted speech.

2.4 Bidirectional Speech Translation (BST)

In the Bidirectional (as opposed to Unidirectional) Translation (MMC-BST) Use Case, 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. If the desired output is speech, users can specify whether their speech features (voice colour, emotional charge, etc.) are preserved in the interpreted speech.

2.5 One-to-Many Speech Translation (MST)

In the One-to-Many (as opposed to Unidirectional or Bidirectional) Translation (MMC-MST) Use Case, 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. If the desired output is speech, users can specify whether their speech features (voice colour, emotional charge, etc.) are preserved in the interpreted speech

2.6 Normative content of the Use Cases

Each Use Case normatively defines:

1. The Functions of the AIW and of the AIMs.
2. The Connections between and among the AIMs
3. The Semantics and the Formats of the input and output data of the AIW and the AIMs.

The word *normatively* implies that an Implementation claiming Conformance to:

1. An *AIW*, shall:
    a. Perform the AIW function specified in the appropriate Section of Chapter 5.
    b. All AIMs, their topology and connections should conform with the AIW Architecture specified in the appropriate Section of Chapter 5.
    c. The AIW and AIM input and output data should have the formats specified in the appropriate Subsection of Section 6.3.
2. An *AIM*, shall:
    a. Perform the AIM function specified by the appropriate section of Chapter 5.
    b. Receive and produce the data specified in the appropriate Subsection of Section 6.1.
    c. Receive as input and produce as output data having the format specified in Section 6.3.
3. A data *Format*, the data shall have the format specified in Section 6.3.

Users of this Technical Specification should note that:

1. This Technical Specification defines Interoperability Levels but does not mandate any.
2. Implementers decide the Interoperability Level 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 Standard.
5. Performance Assessors can assess the level of Performance of an Implementation based on the Performance Assessment specification of this Standard.
6. The MPAI Ecosystem outlined in Annex 2 is governed by [1].
7. Implementers and Users should consider the notices and disclaimers of Annex 3.

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
Degree (of Emotion)	The intensity of an Emotion as one of "Low," "Medium," and "High."
Emotion	An attribute indicating a member of a finite set of Emotions.
Emotion Recognition	An AIM that recognises the best member, or members, of a finite set of Emotions, as conveyed by a Text or Speech Segment or Video.
Face	A video representing human face
Vocal Gesture	Vocal but non-verbal elements of an utterance, e.g., laughs, coughs, hesitation syllables, etc.
Image Analysis	An AIM that extracts pre-specified Features from videos.
Intention	The result of analysis of the goal of an input question.
Language Understanding	An AIM that analyses a natural language Text and returns its Meaning and Emotions.
Meaning	Information – semantic, but also syntactic and structural – extracted from input data, i.e., Text, Speech, and Video.
Object Identifier	A unique ID enabling identification of a physical object, whether animate or inanimate, so that appropriate language and programs can refer to it.
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.
Pitch	The fundamental frequency of Speech. Pitch is the attribute that makes it possible to judge sounds as "higher" and "lower."
Speech Features	Aspects of a speech segment that enable its description and reproduction, e.g., degree of vocal tension, Pitch, etc., and that can be automatically recognised and extracted for speech synthesis or other related purposes.
Speech Rate	The number of Speech Units per second.
Speech Recognition	An AIM that converts Speech to Text.
Speech Synthesis	An AIM that converts Text or concept to Speech.
Speech Unit	Phoneme, syllable, or word as a segment of Speech.
Text	A collection of characters drawn from a finite alphabet.

Text with Emotion	Text marked up with, or associated with, labels or tags indicating emotional or related effects. In this standard, only one emotion is associated with a Text.
Text Translation	An AIM that converts Text in an input language to Text in an output language.

4 References

4.1 Normative References

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

1. MPAI Standard; The governance of the MPAI ecosystem, N341
2. MPAI Technical Specification; AI Framework (MPAI-AIF), N324
3. MPAI Technical Specification: Context-based Audio Enhancement (MPAI-CAE), N326
4. ISO 639; Codes for the Representation of Names of Languages — Part 1: Alpha-2 Code.
5. ISO/IEC 10646; Information technology – Universal Coded Character Set
6. ISO/IEC 14496-10; Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding
7. ISO/IEC 14496-12; Information technology – Coding of audio-visual objects – Part 12: ISO base media file format
8. ISO/IEC 23008-2; Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High Efficiency Video Coding
9. ISO/IEC 23094-1; Information technology – General video coding – Part 1: Essential Video Coding

4.2 Informative References

The references provided here are for information purpose.

1. Ekman, Paul (1999), "Basic Emotions", in Dalgleish, T; Power, M (eds.), Handbook of Cognition and Emotion (PDF), Sussex, UK: John Wiley & Sons
2. Emotion Markup Language (EmotionML) 1.0; https://www.w3.org/TR/2010/WD-emotionml-20100729/diffmarked.html
3. Hobbs J.R., Gordon A.S. (2011) The Deep Lexical Semantics of Emotions. In: Ahmad K. (eds) Affective Computing and Sentiment Analysis. Text, Speech and Language Technology, vol 45. Springer, Dordrecht, https://people.ict.usc.edu/~gordon/publications/EMOT08.PDF and https://www.researchgate.net/publication/227251103_The_Deep_Lexical_Semantics_of_Emotions

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 human's textual and/or vocal utterance in a manner consistent with the human's utterance and emotional state, as detected from the human's text, speech, or face. The machine responds using text, synthetic speech, and a face whose lip movements are synchronised with the synthetic speech.

5.1.2 Input/Output Data

The input and output data of the Conversation with Emotion Use Case are:

Table 2 – I/O Data of Conversation with Emotion

Input	Comments
Input Selection	Data determining the use of Speech vs Text.
Input Text	Text typed by the human as additional information stream or as a replacement of the speech depending on the value of Input Selection.
Input Speech	Speech of the human having a conversation with the machine.
Input Video	Video of the Face of the human having a conversation with the machine.
Output	**Comments**
Output Text	Text of the Speech produced by the machine.
Output Speech	Synthetic Speech produced by the machine.
Output Video	Video of a Face whose lip movements are synchronised with the Output Speech.

5.1.3 Implementation Architecture

The operation of Conversation with Emotion develops as follows:

1. Emotion is recognised in the following way and later reflected in speech production.
    a. A set of Emotion-related cues are extracted from Input Text, Input Speech, and Input Video.
    b. Each AIM (Language Understanding AIM, Speech Recognition AIM, and Video Analysis AIM) independently recognises the Emotion of the input.
    c. The Emotion Fusion AIM fuses all Emotions into the Final Emotion.
2. The Dialog Processing AIM produces a reply based on the Final Emotion and Meaning from the text and video analysis.
3. The Speech Synthesis (Emotion) AIM produces Output Speech from Text with Emotion.
4. The Lips Animation AIM animates the lips of a Face drawn from the Video of Faces Knowledge Base consistently with the Output Speech.

Figure 2 gives the Conversation with Emotion Reference Model including the input/output data, the AIMs and the data exchanged between and among the AIMs.

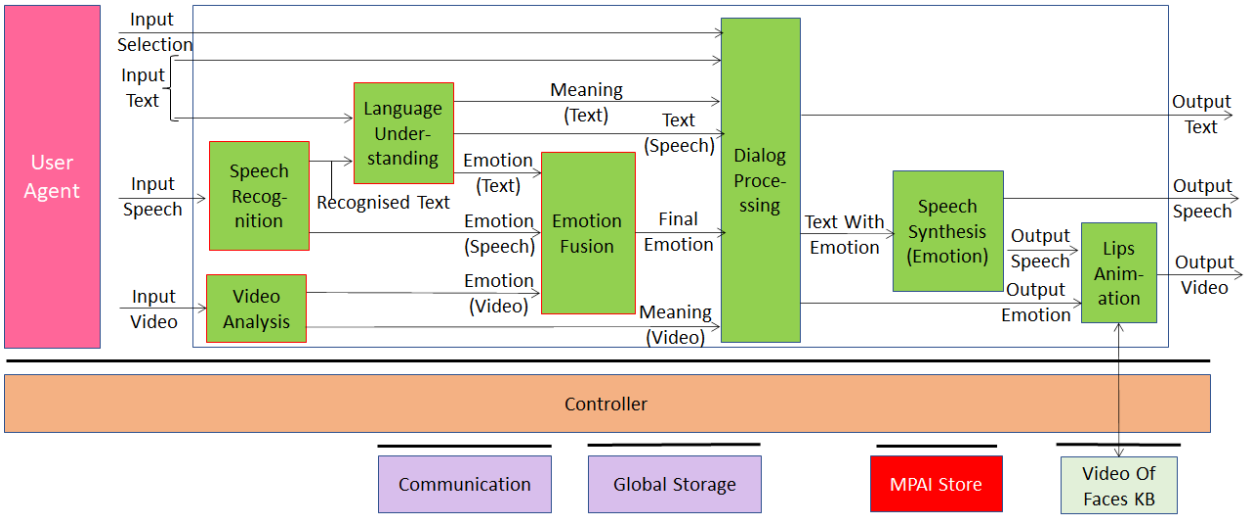

Figure 2 – Reference Model of Conversation With Emotion

5.1.4 AI Modules

The AI Modules of Conversation with Emotion perform the Functions specified in *Table 3*.

Table 3 – AI Modules of Conversation with Emotion

AIM	Function
Language Understanding	Extracts Meaning (Text) and Emotion (Text) included in Input Text containing natural language.
Speech Recognition	Analyses Input Speech and produces Recognised Text and Emotion (Speech).
Video Analysis	Extracts Meaning (Video) and Emotion (Video) carried by Input Video.
Emotion Fusion	Determines the Final Emotion from Emotion (Text), Emotion (Speech), and Emotion (Video).
Dialog Processing	Produces Output Emotion and Output Speech by analysing Meaning (Text), Meaning (Video), and Text (Speech) or Input Text, depending on the value of Input Selection.
Speech Synthesis	Produces Output Speech from Text with Emotion.
Lips Animation	Animates the lips of a video obtained by querying the Video Faces KB, using the Output Emotion.

5.1.5 AIW Metadata

Specified in Annex 4 Section 1.

5.2 Multimodal Question Answering (MQA)

5.2.1 Scope of standard

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

5.2.2 Input/output data

The input and output data of the Multimodal Question Answering Use Case are:

Table 4 – I/O Data of Multimodal Question Answering

Input	Comments
Input Selection	Data determining the use of Speech or Text.
Input Text	Text typed by the human as a replacement for Input Speech.
Input Speech	Speech of the human asking the machine a question.
Input Video	Video of the human showing a held object.
Output	**Comments**
Output Text	The Text generated by MQA in response to human inputs.
Output Speech	The Speech generated by MQA in response to human inputs.

5.2.3 Implementation Architecture

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

1. Human asks a question in the form of text or speech depending on the value of the Input Selection.

2. Language Understanding extracts the Meaning of the question from Input Text or Input Speech depending on the value of Input Selection.
3. Video Analysis identifies the object.
4. Question Analysis determines the Intention of the question.
5. Question Answering uses Intention and Meaning to produce the answer as Reply Text.
6. Speech Synthesis (Text) produces the Output Speech from Reply Text.

Figure 3 depicts the input/output data, the AIMs and the data exchanged between the AIMs.

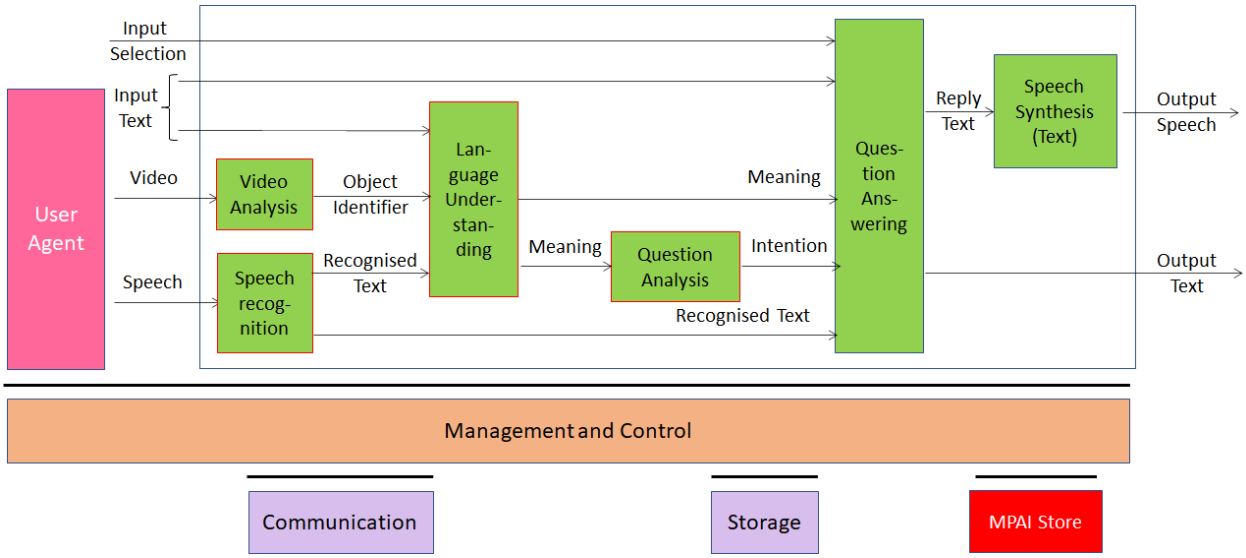

Figure 3 – Reference Model of Multimodal Question Answering

5.2.4 AI Modules

The AI Modules of Multimodal Question Answering are given in *Table 5*.

Table 5 – AI Modules of Multimodal Question Answering

AIM	Function
Video Analysis	Produces the name of the object in focus by analysing Input Video.
Speech Recognition	Produces Recognised Text by recognising Input Speech.
Language Understanding	Produces Meaning by analysing Input Text or Input Speech depending on the value of Input Selection.
Question Analysis	Determines Intention by analysing Meaning.
Question Answering	Produces Reply Text by analysing Input Text or Recognised Text depending on the value of Input Selection, Intention and Meaning.
Speech Synthesis (Text)	Converts Reply Text to Output Speech.

5.2.5 AIW Metadata

Specified in Annex 5 Section 1.

5.3 Unidirectional Speech Translation (UST)

5.3.1 Scope of Use Case

In Unidirectional Speech Translation (UST), Speech in Language A are translated into Speech in Language B. The flow of control is from Input Speech or Input Text to Translated Text, and then to Output Speech and Output Text. Depending on the value of Input Selection

1. Input Text in Language A is translated into Translated Text in Language B and pronounced as Speech in Language B.
2. The Speech features (voice colour, emotional charge, etc.) in Language A are preserved in the Language B.

5.3.2 Input/output data

The input and output data of the Unidirectional Speech Translation Use Case are:

Table 6 – I/O Data of Unidirectional Speech Translation

Input	Comments
Input Selection	Determines whether 1. The input will be in Text or Speech 2. The Input Speech features are preserved in the Output Speech.
Requested Languages	User-specified input Language (A) and output Language (B).
Input Speech	Speech produced in Language A by a human desiring translation into language B.
Input Text	Alternative textual source information to be translated into and pronounced in language B depending on the value of Input Selection.
Output	**Comments**
Translated Speech	Input Speech translated into language B preserving the Input Speech features in the Output Speech, depending on the value of Input Selection.
Translated Text	Text of Input Speech or Input Text translated into language B, depending on the value of Input Selection.

5.3.3 Implementation Architecture

Figure 4 describes the input/output data, the AIMs and the data exchanged between AIMs.

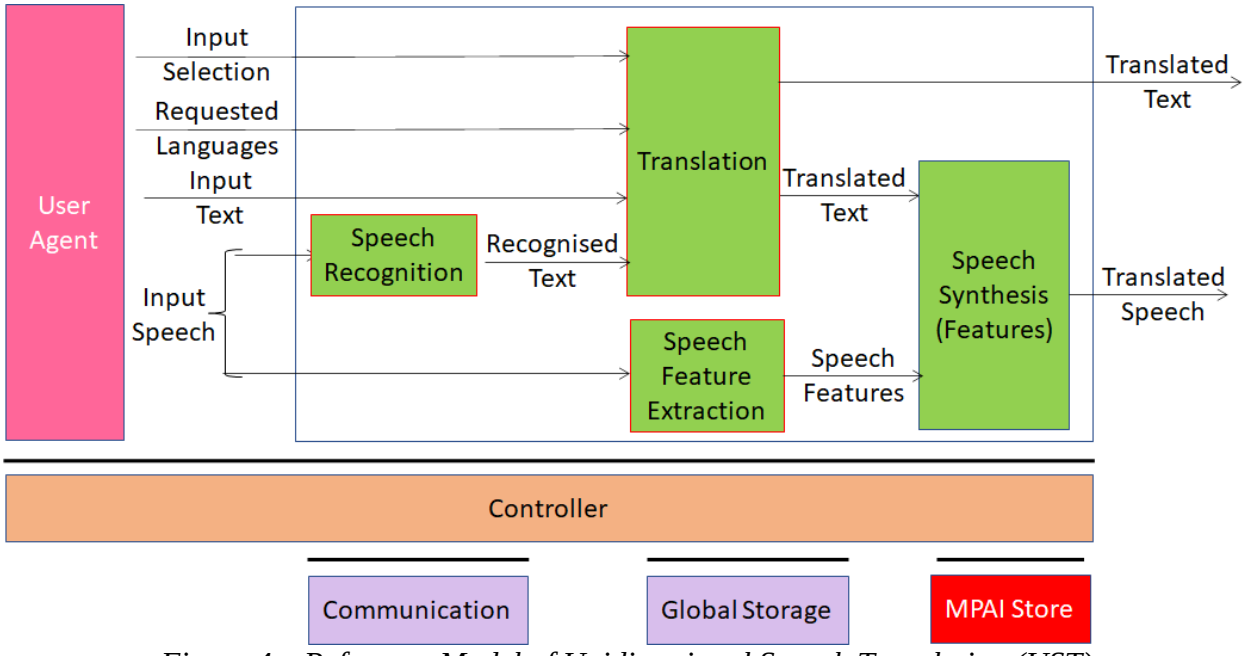

Figure 4 – Reference Model of Unidirectional Speech Translation (UST)

5.3.4 AI Modules

The AI Modules of Unidirectional Speech Translation are given in *Table 7*.

Table 7 – AI Modules of Unidirectional Speech Translation

AIM	Function
Speech Recognition	Converts Input Speech into Recognised Text.
Translation	Translates Input Text or Recognised Text (depending on the value of Input Selection) into the target language.
Speech Feature Extraction	Extracts Speech Features specific to the speaker (e.g., tones, intonation, intensity, pitch, emotion, speed) from Input Speech.
Speech Synthesis (Features)	Produces Output Speech from Translated Text using the Speech Features, depending on the value of Input Selection.

5.3.5 AIW Metadata

Specified in Annex 6 Section 1.

5.4 Bidirectional Speech Translation (BST)

5.4.1 Scope of Use Case

In Bidirectional (as opposed to Unidirectional) Speech Translation, two people converse, one speaking language A and the other language B. The flow of control (from Input Speech to Translated Text to Output Speech) is identical to that of the Unidirectional case. The difference is that, rather than one such flow, two flows are provided in two different channels – the first from language A to language B, and the second from language B to language A.

Depending on the value of Input Selection

1. Input Text in Language A is translated into Translated Text in Language B and pronounced as Speech in Language B.
2. The Speech features (voice colour, emotional charge, etc.) in Language A are preserved in Language B.

The same applies for the Language-B-to-Language-A channel.

5.4.2 Input/output data

The input and output data of the Bidirectional Speech Translation Use Case are:

Table 8 – I/O Data of Bidirectional Speech Translation

Input	Comments
Input Selection	Determines whether the input will be Text or Speech.
Requested languages	User-specified input language and output languages
Input Speech1	Speech by human1 desiring spoken translation in the specified language.
Input Text1	Alternative Input Text to be translated to the specified language.
Input Speech2	Speech by human2 desiring spoken translation in the specified language.
Input Text2	Alternative Input Text to be translated to the specified language.
Output	**Comments**
Output Speech1	Translated Speech of Speaker 1.
Output Text1	Text of the translated Speech of Speaker 1.
Output Speech2	Translated Speech of Speaker 2.
Output Text2	Text of the translated Speech of Speaker 2.

5.4.3 Implementation Architecture

Figure 5 depicts the AIMs and the data exchanged between AIMs.

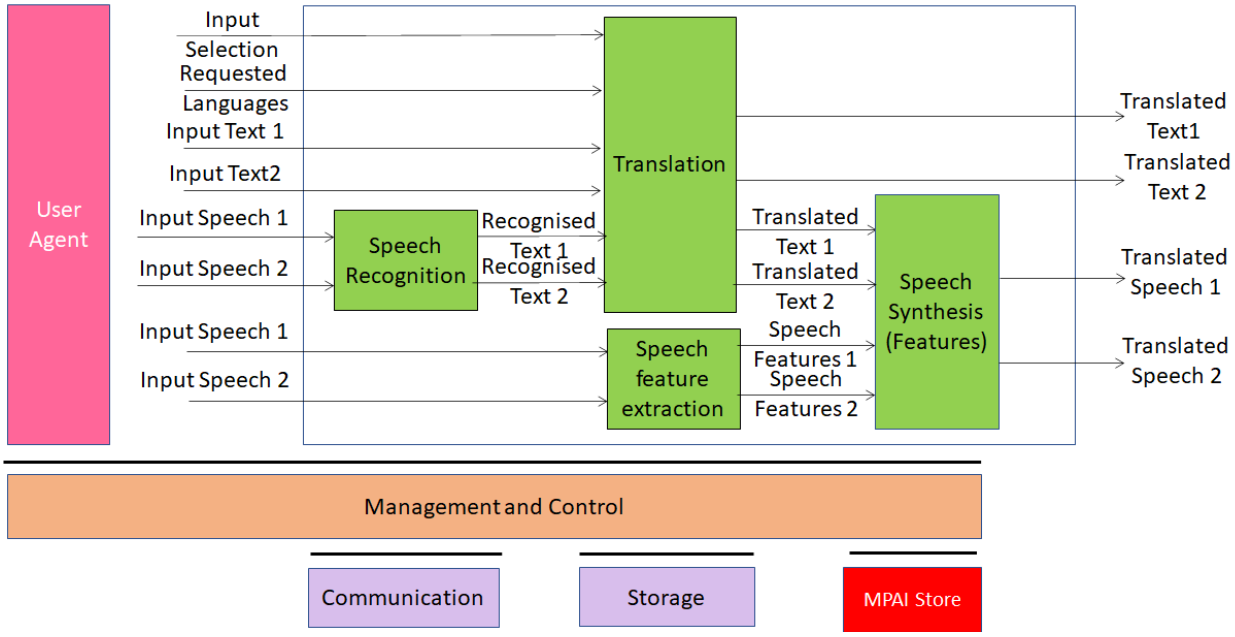

Figure 5 – Reference Model of Bidirectional Speech Translation (BST)

5.4.4 AI Modules

The AI Modules are given in *Table 9*.

Table 9 – AI Modules of Bidirectional Speech Translation

AIM	Function
Speech Recognition	Converts Input Speech 1 and Input Speech 2 into Recognised Text 1 and Recognised Text 2.
Translation	Translates Input Text1 and Input Text 2 (or Recognised Text 1 and Recognised Text 2) – depending on the value of Input Selection – into Translated Text 1 and Translated Text 2.
Speech Feature Extraction	Extracts Speech Features 1 and Speech Features 2 from Input Speech 1 and Input Speech 2.
Speech Synthesis (Features)	Produces Translated Speech 1 and Translated Speech 2 from Translated Text 1 and Translated Text 2, making use of the Speech Features extracted from the corresponding Input Speech segments, depending on the value of Input Selection.

5.4.5 AIW Metadata

Specified in Annex 7 Section 1.

5.5 One-to-Many Speech Translation (MST)

5.5.1 Scope of Use Case

In One-to-Many (as opposed to Unidirectional or Bidirectional) Speech Translation, any person speaking his or her preferred language broadcasts to two or more audience members, each listening, in a different language. The flow of control (from Recognised Text to Translated Text to Output Speech) 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.
Depending on the value of Input Selection (text or speech):

1. Input Text in Language A is translated into Translated Text in and pronounced as Speech of all Requested Languages.
2. The Speech features (voice colour, emotional charge, etc.) in Language A are preserved in all Requested Languages.

5.5.2 Input/output data

The input and output data of the One-to-Many Speech Translation Use Case are:

Table 10 – I/O Data of One-to-Many Speech Translation

Input	Comments
Input Selection	Determines whether the input will be in Text or Speech.
Desired Languages	User-specified input language and translated languages
Input Speech	Speech produced by human desiring translation and interpretation in a specified set of languages.
Input Text	Alternative textual source information.
Output	**Comments**
Translated Speech	Speech translated into the Desired Languages.
Translated Text	Text translated into the Desired Languages.

5.5.3 Implementation Architecture

Figure 6 depicts the AIMs and the data exchanged between AIMs.

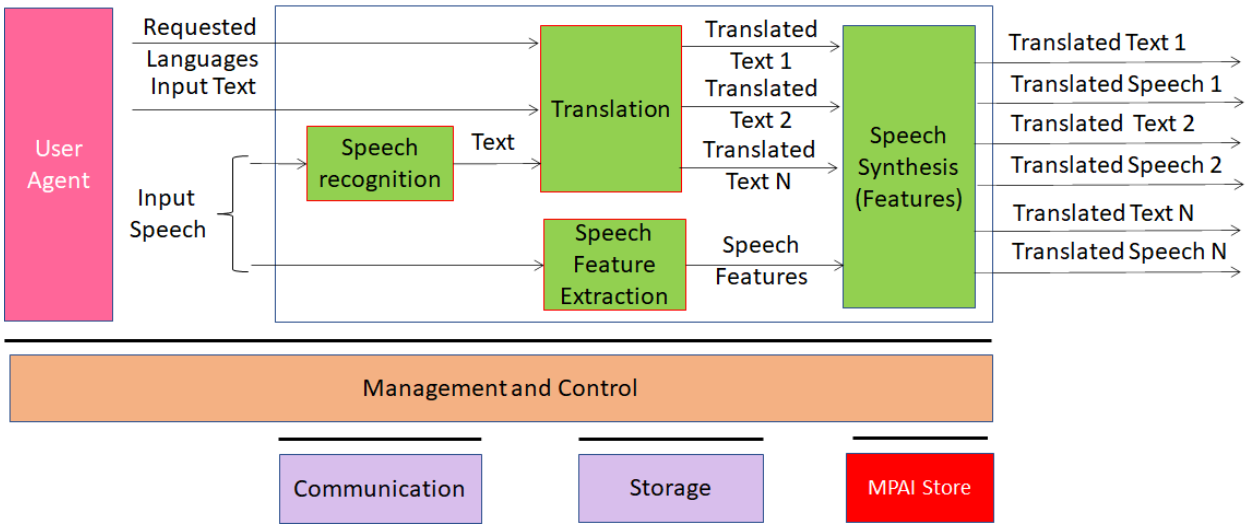

Figure 6 – Reference Model of One-to-Many Speech Translation (MST)

5.5.4 AI Modules

The AI Modules of Personalised Automatic Speech Translation are given in *Table 11*.

Table 11 – AI Modules of One-to-Many Speech Translation

AIM	Function
Speech Recognition	Converts one Input Speech Segment into Recognised Text

Speech Feature Extraction	Extracts speaker-specific Speech Features from the Input Speech.
Translation	Translates one Text input into a set of Translated Texts in the Requested Languages.
Speech Synthesis (Features)	Uses the set of Translated Texts and the Speech Features to produce a set of Speech Segments in the Desired Languages.

5.5.5 AIW Metadata

Specified in Annex 8 Section 1.

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 using the AIM Metadata specified by the AI Framework (MPAI-AIF) Standard [2].

Section 6.3 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 [3]. However, 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 AI Module Interoperability

To the extent possible, AIM input and output data are specified in a way that is neutral to the technology used to implement the AIM internals. In some cases, however, AIM input and output data of strongly depend on whether the technology used is data processing or Artificial Intelligence. If an AIM is based on, e.g., a neural network, it will need either (1) a usable neural model whose training has included specifiable features, or (2) a precise specification of the features themselves plus an adequate training corpus, so that the AIM using that data can create its own usable model.

6.2 MPAI-MMC AIMs and their data

6.2.1 Conversation with Emotion (CWE)

Table 12 gives the AIMs and the input/output data of the Conversation with Emotion Use Case.

Table 12 – Conversation with Emotion AIMs and data formats

AIM	Input Data	Output Data
Video analysis	Input Video	Emotion (Video) Meaning (Video)
Speech recognition	Input Speech	Recognised Text Emotion (Speech)
Language understanding	Input Text Recognised Text	Meaning (Text) Text (Speech) Emotion (Text)
Emotion Fusion	Emotion (Text) Emotion (Speech) Emotion (Video)	Final Emotion

Dialog processing	Input Selection Input Text Meaning (Text) Text (Speech) Final Emotion Meaning (Speech) Meaning (Video)	Output Text Text with Emotion Output Emotion
Speech Synthesis (Emotion)	Text with Emotion	Output Speech
Lips animation	Output Speech Final Emotion Video of Face	Output Video

The AIM Metadata are given in Annex 4 Section 2.

6.2.2 Multimodal Question Answering (MQA)

Table 13 gives the AIMs and the input/output data of the Multimodal Question Answering Use Case.

Table 13 – Multimodal Question Answering AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Input Speech	Recognised Text
Video Analysis	Video	Object Identifier
Language understanding	Input Text Recognised Text	Meaning Meaning
Question analysis	Meaning	Intention
Question Answering	Input Selection Input Text Meaning Intention Recognised Text	Output Text
Speech Synthesis (Text)	Output Text	Output Speech

The AIM Metadata are given in Annex 5 Section 2.

6.2.3 Unidirectional Speech Translation (UST)

Table 14 gives the AIMs and the input/output data of the Unidirectional Speech Translation Use Case.

Table 14 – Unidirectional Speech Translation AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Input Speech	Recognised Text
Translation	Input Selection Requested Language Input Text	Translated Text
Speech feature extraction	Input Speech	Speech Features
Speech synthesis (Features)	Translated Text Speech features	Translated Speech

The AIM Metadata are given in Annex 6 Section 2.

6.2.4 Bidirectional Speech Translation (BST)

Table 15 gives the AIMs and the input/output data of the Bidirectional Speech Translation Use Case.

Table 15 – Bidirectional Speech Translation AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Input Speech 1 Input Speech 2	Recognised Text 1 Recognised Text 2
Translation	Input Selection Requested Languages Input Text 1 Input Text 2	Translation Text1 Translation Text2
Speech feature extraction	Input Speech 1 Input Speech 2	Speech Features 1 Speech Features 2
Speech synthesis (Features)	Translation Text1 Translation Text2 Speech Features 1 Speech Features 2	Output Speech 1 Output Speech 2

The AIM Metadata are given in Annex 7 Section 2.

6.2.5 One-to-Many Speech Translation (MST)

Table 16 gives the input/output data of the One-to-many Speech Translation Use Case.

Table 16 – One-to-many Speech Translation AIMs and data formats

AIM	Input Data	Output Data
Speech Recognition	Input Speech	Recognised Text
Translation	Requested languages Input Text Input Speech	Translated Text 1 ... Translated Text N
Speech feature extraction	Input Speech	Speech Features
Speech synthesis (Features)	Text (Translation result) Speech Features	Translated Text 1 Translated Speech 1 ... Translated Text N Translated Speech N

The AIM Metadata are given in Annex 4 Section 8.

6.3 Data Formats

The data Formats specified in this Technical Specification are listed in

Table 17. 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 17 – Data formats

Name of Data Format	Subsection	Use Case
Emotion	6.3.1	CWE
Intention	6.3.2	MQA
Language identifier	6.3.3	UST
		BST
		MST
Meaning	6.3.4	CWE
Object identifier	6.3.5	MQA
Speech	6.3.6	CWE
		MQA
		UST
		BST
		MST
Speech Features	6.3.7	UST
	6.3.7	UST
	6.3.7	UST
Text	6.3.8	CWE
		MQA
		UST
		BST
		MST
Text with Emotion	6.3.9	CWE
Video	6.3.10	CWE
Video File	6.3.11	
Video Of Faces KB Query Format	6.3.12	CWE

6.3.1 Emotion

The Syntax and Semantics of Emotion (Text), Emotion (Speech), Emotion (Video), Final Emotion and Output Emotion are given by the following clauses.

6.3.1.1 Syntax

Human Emotion is represented by.

```
{
  "$schema":"http://json-schema.org/draft-07/schema",
  "definitions":{
    "EmotionType":{
      "type":"object",
      "properties":{
        "emotionDegree":{
          "type":"{Enum high | Enum medium | Enum low}"
        },
        "emotionName":{
          "type":"string"
        },
```

```
        "emotionSetName":{
          "type":"string"
        }
      }
    },
    "type":"object",
    "properties":{
      "primary":{
        "$ref":"#/definitions/EmotionType"
      },
      "secondary":{
        "$ref":"#/definitions/EmotionType"
      }
    }
  }
```

6.3.1.2 Semantic

Name	*Definition*
`EmotionType`	Specifies the Emotion that the input carries.
`emotionDegree`	Specifies the Degree of Emotion as one of "Low," "Medium," and "High."
`emotionName`	Specifies the name of an Emotion.
`emotionSetName`	Specifies the name of the Emotion set which contains the Emotion. Emotion set of *Table 18* is used as a baseline, but other sets are possible.

Emotions are expressed vocally through combinations of prosody (pitch, rhythm, and volume variations); separable speech effects (such as degrees of voice tension, breathiness, etc.); and vocal gestures (laughs, sobs, etc.).
Table 18 gives the MPAI standardised three-level Emotion set partly based on Paul Eckman [1]. The EMOTION CATEGORIES column specifies the categories using nouns; the GENERAL ADJECTIVAL column gives adjectival labels for general or basic emotions within a category; and the SPECIFIC ADJECTIVAL column gives labels for more specific (sub-categorized) emotions in the relevant category, often (but not always) representing differing degrees of the basic emotion.

Table 19 provides the semantics for each label in the GENERAL ADJECTIVAL and SPECIFIC ADJECTIVAL columns.
An Implementer wishing to extend or replace *Table 18* is requested to do the following:
1. Create a new *Table 18* where
    a. Proposed additions are clearly marked (in case of extension)
    b. All Emotions and levels (up to 3) are listed (in case of replacement).
2. Create a new
3. *Table 19* where the semantics of the Emotions is
    a. added to the semantics of the existing emotions (in case of extension)
    b. is provided (in case of replacement).

   The semantics provided should have a level of details comparable to the semantics given in the current

Table 19.

4. Submit both tables to the MPAI Secretariat.

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

The accepted new tables will be posted to the MPAI web site.

Table 18 – Basic Emotions

EMOTION CATEGORIES	GENERAL ADJECTIVAL	SPECIFIC ADJECTIVAL
HAPPINESS	happy	joyful content delighted amused
SADNESS	sad	lonely grief-stricken discouraged depressed disappointed
CALMNESS	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	sceptical
AROUSAL	aroused/excited/energetic	cheerful playful lethargic sleepy

Table 19 – 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
sceptical	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

6.3.2 Intention

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

- qtopic
- qfocus
- qLAT
- qSAT

6.3.2.1 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" }
  }
}
```

6.3.2.2 Semantics

Name	*Definition*
`Intention`	Provides abstracts of Intention of User Question using properties: qtopic, qfocus, qLAT, qSAT and qdomain
`qtopic`	Indicates the topic of the question. Question topic is the object or event that the question is about. Ex. of Qtopic is King Lear in "Who is the author of King Lear?".
`qfocus`	Indicates 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 "Biden" in the Answer.) Answer: Biden is the president of USA.
`qLAT`	Indicates the lexical answer type of the question.
`qSAT`	Indicates the semantic answer type of the question. QSAT corresponds to Named Entity type of the language analysis results.
`qdomain`	Indicates 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", and 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 "
    }
  ]
}
```

6.3.3 Language identifier

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

6.3.4 Meaning

This subclause specifies data formats to describe meaning which is the result of natural language analysis. The “meaning” consists of the following elements.

- POS_tagging
- NE_tagging
- Dependency_tagging
- SRL_tagging

6.3.4.1 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"
      }
    }
  }
}
```


6.3.4.2 Semantics

Name	*Definition*
`Meaning`	Provides an abstract of description of natural language analysis results.
`POS_tagging`	Indicates 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`	Indicates NE tagging results including information on the NE tagging set and tagged results of the User question. NE: Named Entity such as Person, Organisation, Fruit, etc.
`dependency_tagging`	Indicates 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`	Indicates 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.

6.3.5 Object Identifier

An object is identified as follows.

6.3.5.1 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"
    }
  }
}
```

6.3.5.2 Semantics

Name	*Definition*
`objectIdentifier`	Provides the description of the recognised object.
`objectImageLabel`	Indicates the recognised object's label in the object image.
`confidenceLevel`	Indicates the confidence level of the object image label recognised by the "Video analysis".

6.3.6 Speech

Digital representation of analogue speech sampled at a frequency between 8 kHz and 96 kHz with a number of bits between 16 bits/sample and 24 bits/sample PCM values.

6.3.7 Speech Features

Speech Features are digitally represented as follows.

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

```
        "properties":{
          "pitch":{
            "type":"real"
          },
          "tone":{
            "type":"ToneType"
          },
          "intonation":[
            {
              "type_p":"pitch",
              "type_s":"speed",
              "type_i":"intensity"
            }
          ],
          "intensity":{
            "type":"real"
          },
          "speed":{
             "type":"real",
          },
          "emotion":{
            "type":"EmotionType"
          },
          "NNSpeechFeatures":{
            "type":"vector of floating point"
          }
        }
      }
    },
    "type":"object",
    "properties":{
      "primary":{
        "$ref":"#/definitions/SpeechFeatureType"
      },
      "secondary":{
        "$ref":"#/definitions/SpeechFeatureType"
      }
    }
}

{
    "$schema":"http://json-schema.org/draft-07/schema",
    "definitions":{
      "ToneType":{
        "type":"object",
        "properties":{
          "toneName":{
            "type":"string"
          },
          "toneSetName":{
```

```
            "type":"string"
          }
        }
      },
      "type":"object",
      "properties":{
        "primary":{
          "$ref":"#/definitions/ToneType"
        },
        "secondary":{
          "$ref":"#/definitions/ToneType"
        }
      }
    }
}
```

6.3.7.1 Semantics

Name	*Definition*
SpeechFeatures	Indicates characteristic elements extracted from the input speech, specifically pitch, tone, intonation, intensity, speed, emotion, and NNspeechFeatures.
NNSpeechFeatures	Indicates specifically neural-network-based characteristic elements extracted from the input speech by Neural Network
pitch	Indicates the fundamental frequency of Speech expressed as a real number indicating frequency as Hz (Hertz).
tone	Tone is a variation in the pitch of the voice while speaking expressed as human readable words as in *Table 18*.
ToneType	Indicates the Tone that the input speech carries.
intonation	A variation of the pitch, intensity and speed within a time period measured in seconds.
intensity	Energy of Speech expressed as a real number indicating dBs (decibel).
speed	Indicates the Speech Rate as a real number indicating specified linguistic units (e.g., Phonemes, Syllables, or Words) per second.
emotion	Indicates the Emotion that the input speech carries.
EmotionType	Indicates the Emotion that the input speech carries.
toneName	Specifies the name of a Tone.
toneSetName	Name of the Tone set which contains the Tone. Tone set is used as a baseline, but other sets are possible.

Note: The semantics of "tone" defines a basic set of elements characterising tone. Elements can be added to the basic set or new sets defined using the registration procedure defined for Emotion Sets (6.3.1).

Table 20 – Basic Tones

TONE CATEGORIES	ADJECTIVAL	Semantics
FORMALITY	formal informal	serious, official, polite everyday, relaxed, casual
ASSERTIVENESS	assertive factual hesitant	certain about content neutral about content uncertain about content
REGISTER (per situation or use case)	conversational directive	appropriate to an informal speaking related to commands or requests for action

6.3.8 Text

The Format of Input Text, Output Text and Recognised Text is provided by ISO/IEC 10646; Information technology – Universal Coded Character Set [5].

6.3.9 Text with Emotion

Text With Emotion is digitally represented as follows.

6.3.9.1 Syntax

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

6.3.9.2 Semantics

Name	*Definition*
`TextWithEmotionType`	Indicates the Emotion that the text carries.
`emotionDegree`	Indicates the Degree of the Emotion expressed as human readable words: "Low", "Medium", "High".

Name	*Definition*
`emotionName`	Indicates the name of the Emotion.
`emotionSetName`	Name of the Emotion Set which used to describe the Emotion: Basic, Extended or Proprietary Emotion Set (6.3.1).

6.3.10 Video

Video satisfies the following specifications:

1. Pixel shape: square
2. Bit depth: 8 or 10 bits/pixel
3. Aspect ratio: 4/3 or 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 or BT2020
9. Colour format: RGB or YUV
10. Compression:
    a. uncompressed;
    b. if compressed, compression according to one of the following standards: MPEG-4 AVC [6], MPEG-H HEVC [8], MPEG-5 EVC [9]

6.3.11 Video File

The Format of a Video MP4 File Format [7].

6.3.12 Video of Faces KB Query Format

Data Specification: All faces in the Video of Faces KB must be aligned.
Input: The Video of Faces KB is queried with an Emotion.
Output: The response is a Video File of a human face..

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 21*.

Table 21 – MPAI-wide Terms

Term	Definition
Access	Static or slowly changing data that are required by an application such as domain knowledge data, data models, etc.
AI Framework (AIF)	The environment where AIWs are executed.
AI Module (AIM)	A data processing element receiving AIM-specific Inputs and producing AIM-specific Outputs according to according to its Function. An AIM may be an aggregation of AIMs.
AI Workflow (AIW)	A structured aggregation of AIMs implementing a Use Case receiving AIM-specific inputs and producing AIM-specific inputs according to its Function.
Application Standard	An MPAI Standard designed to enable a particular application domain.
Channel	A connection between an output port of an AIM and an input port of an AIM. The term "connection" is also used as synonymous.
Communication	The infrastructure that implements message passing between AIMs
Component	One of the 7 AIF elements: Access, Communication, Controller, Internal Storage, Global Storage, MPAI Store, and User Agent
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 Means to Test the Conformance of an Implementation.
Conformance Testing Means	Procedures, tools, data sets and/or data set characteristics to Test the Conformance of an Implementation.
Connection	A channel connecting an output port of an AIM and an input port of an AIM.
Controller	A Component that manages and controls the AIMs in the AIF, so that they execute in the correct order and at the time when they are needed
Data Format	The standard digital representation of data.
Data Semantics	The meaning of data.
Ecosystem	The ensemble of the following actors: 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 operations effected by an AIW or an AIM on input data.
Global Storage	A Component to store data shared by AIMs.
Internal Storage	A Component to store data of the individual AIMs.

Identifier	A name that uniquely identifies an Implementation.
Implementation	1. An embodiment of the MPAI-AIF Technical Specification, or 2. An AIW or AIM of a particular Level (1-2-3) conforming with a Use Case of an MPAI Application Standard.
Interoperability	The ability to functionally replace an AIM with another AIM having the same Interoperability Level
Interoperability Level	The attribute of an AIW and its AIMs to be executable in an AIF Implementation and to: 1. Be proprietary (Level 1) 2. Pass the Conformance Testing (Level 2) of an Application Standard 3. `Pass the Performance Testing (Level 3) of an Application Standard.
Knowledge Base	Structured and/or unstructured information made accessible to AIMs via MPAI-specified interfaces
Message	A sequence of Records transported by Communication through Channels.
Normativity	The set of attributes of a technology or a set of technologies specified by the applicable parts of an MPAI standard.
Performance	The attribute of an Implementation of being Reliable, Robust, Fair and Replicable.
Performance Assessment	The normative document specifying the 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 used in MPAI-AIF or an AIW of an Application Standard and, where applicable, the classes, other subsets, options and parameters relevant to that subset.
Record	A data structure with a specified structure
Reference Model	The AIMs and theirs Connections in an AIW.
Reference Software	A technically correct software implementation of a Technical Specification containing source code, or source and compiled code.
Reliability	The attribute of an Implementation that performs as specified by the Application Standard, profile and version the Implementation refers to, e.g., within the application scope, stated limitations, and for the period of time specified by the Implementer.
Replicability	The attribute of an Implementation whose Performance, as Assessed by a Performance Assessor, can be replicated, within an agreed level, by another Performance Assessor.
Robustness	The attribute of an Implementation that copes with data outside of the stated application scope with an estimated degree of confidence.
Scope	The domain of applicability of an MPAI Application Standard
Service Provider	An entrepreneur who offers an Implementation as a service (e.g., a recommendation service) to Users.
Standard	The ensemble of Technical Specification, Reference Software, Conformance Testing and Performance Assessment of an MPAI application Standard.
Technical Specification	(Framework) the normative specification of the AIF.

	(Application) the normative specification of the set of AIWs belonging to an application domain along with the AIMs required to Implement the AIWs that includes: 1. The formats of the Input/Output data of the AIWs implementing the AIWs. 2. The Connections of the AIMs of the AIW. 3. The formats of the Input/Output data of the AIMs belonging to the AIW.
Testing Laboratory	A laboratory accredited by MPAI to Assess the Grade of Performance of Implementations.
Time Base	The protocol specifying how Components can access timing information
Topology	The set of AIM Connections of an AIW.
Use Case	A particular instance of the Application domain target of an Application Standard.
User	A user of an Implementation.
User Agent	The Component interfacing the user with an AIF through the Controller
Version	A revision or extension of a Standard or of one of its elements.

Annex 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.

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

Life cycle of MPAI Standards

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

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

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

Coverage and Applicability of MPAI Standards

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

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

IN NO EVENT SHALL MPAI BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND 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. Submitters of technologies to this standard have agreed to licence their Intellectual Property according to their respective Framework Licences.

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 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.

Copyright
MPAI draft and approved standards, whether they are in the form of documents or as web pages or otherwise, are copyrighted by MPAI under Swiss and international copyright laws. MPAI Standards are made available and may be used for a wide variety of public and private uses, e.g., implementation, use and reference, in laws and regulations and standardisation. By making these documents available for these and other uses, however, MPAI does not waive any rights in copyright to its Standards. For inquiries regarding the copyright of MPAI standards, please contact the MPAI Secretariat.

The Reference Software of an MPAI Standard is released with the MPAI Modified Berkeley Software Distribution licence. 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)

Level 1 Interoperability

With reference to *Figure 1*, MPAI issues and maintains a standard – called MPAI-AIF – whose components are:

1. An environment called AI Framework (AIF) running AI Workflows (AIW) composed of inter-connected AI Modules (AIM) exposing standard interfaces.
2. A distribution system of AIW and AIM Implementation called MPAI Store from which an AIF Implementation can download AIWs and AIMs.

Implementers' benefits	Upload to the MPAI Store and have globally distributed Implementations of - AIFs conforming to MPAI-AIF. - AIWs and AIMs performing proprietary functions executable in AIF.
Users' benefits	Rely on Implementations that have been tested for security.
MPAI Store's role	- Tests the Conformance of Implementations to MPAI-AIF. - Verifies Implementations' security, e.g., absence of malware. - Indicates unambiguously that Implementations are Level 1.

Level 2 Interoperability

In a Level 2 Implementation, the AIW must be an Implementation of an MPAI Use Case and the AIMs must conform with an MPAI Application Standard.

Implementers' benefits	Upload to the MPAI Store and have globally distributed Implementations of - AIFs conforming to MPAI-AIF. - AIWs and AIMs conforming to MPAI Application Standards.
Users' benefits	- Rely on Implementations of AIWs and AIMs whose Functions have been reviewed during standardisation. - Have a degree of Explainability of the AIW operation because the AIM Functions and the data Formats are known.
Market's benefits	- Open AIW and AIM markets foster competition leading to better products. - Competition of AIW and AIM Implementations fosters AI innovation.
MPAI Store's role	- Tests Conformance of Implementations with the relevant MPAI Standard. - Verifies Implementations' security. - Indicates unambiguously that Implementations are Level 2.

Level 3 Interoperability

MPAI does not generally set standards on how and with what data an AIM should be trained. This is an important differentiator that promotes competition leading to better solutions. 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.

Implementers' benefits	May claim their Implementations have passed Performance Assessment.
Users' benefits	Get assurance that the Implementation being used performs correctly, e.g., it has been properly trained.
Market's benefits	Implementations' Performance Grades stimulate the development of more Performing AIM and AIW Implementations.

MPAI Store's role	-	Verifies the Implementations' security
	-	Indicates unambiguously that Implementations are Level 3.

The MPAI ecosystem

The following *Figure 7* 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. Implementers submit Implementations to Performance Assessors (step 4).
5. If the Implementation Performance is acceptable, Performance Assessors inform Implementers (step 5a) and MPAI Store (step 5b).
6. Implementers submit Implementations to the MPAI Store (step 6); The Store Tests Conformance and security of the Implementation.
7. Users download Implementations (step 7).

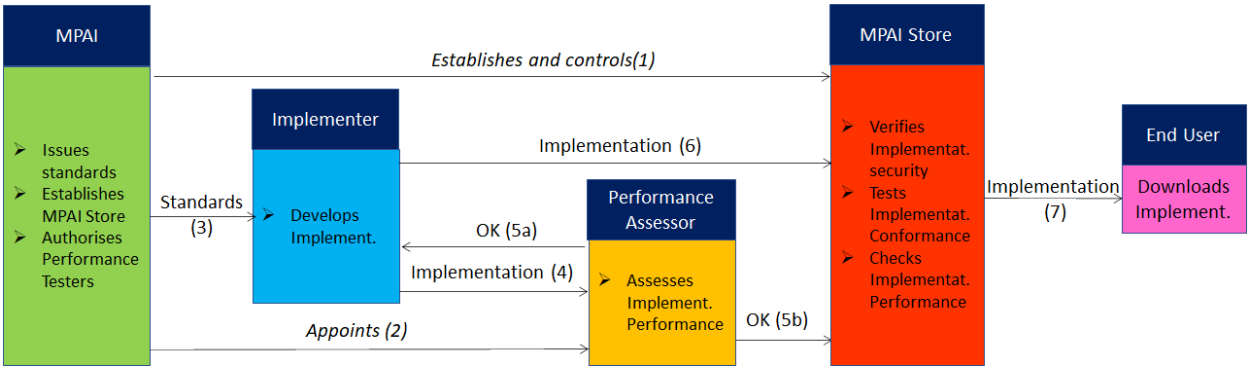

Figure 7 – The MPAI ecosystem operation

Annex 4 – AIW and AIM Metadata of MMC-CWE

1 AIW metadata for CWE


```
{
   "AIM":{
      "Implementer_ID":"*",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"CWE",
         "Version":"1",
         "aName":"_MAIN_"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIW implements the CWE Use Case of MPAI-MMC",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "Video_t":"uint24[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"{Enum Text | Enum Speech}",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputVideo",
            "Direction":"InputOutput",
            "Record_Type":"Video_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputSpeech",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OuputVideo",
            "Direction":"OutputInput",
            "Record_Type":"Video_t",
```

```
      "Type":"Software",
      "Protocol":""
   }
],
"AIMs":[
   {
      "SpeechRecogniton":"@*:(S:(MMC:CWE:2:SpeechRecogniton)):*",
      "VideoAnalysis":"@*:(S:(MMC:CWE:2:VideoAnalysis)):12",
      "LanguageUnderstanding":"@*:(S:(MMC:CWE:2:LanguageUnderstanding)):*",
      "EmotionFusion":"@*:(S:(MMC:CWE:2: EmotionFusion)):*",
      "DialogProcessing":"@*:(S:(MMC:CWE:2:DialogProcessing)):*",
      "SpeechSynthesis":"@*:(S:(MMC:CWE:2:SpeechSynthesis)):*",
      "LipsAnimation":"@*:(S:(MMC:CWE:2:LipsAnimation)):*"
   }
],
"Topology":[
   {
      "InputSelection":{
         "Output":{
            "Module":"",
            "Port":"InputSelection"
         },
         "Input":{
            "Module":"DialogueProcessing",
            "Port":"InputSelection"
         }
      },
      "InputText1":{
         "Output":{
            "Module":"",
            "Port":"InputText"
         },
         "Input":{
            "Module":"LanguageUnderstanding",
            "Port":"InputText"
         }
      },
      "InputText2":{
         "Output":{
            "Module":"",
            "Port":"InputText"
         },
         "Input":{
            "Module":"LanguageUnderstanding",
            "Port":"InputText"
         }
      },
      "Text(Speech)":{
         "Output":{
            "Module":"LanguageUnderstanding",
            "Port":"Text(Speech)"
         },
         "Input":{
            "Module":"DialogProcessing",
            "Port":"Text(Speech)"
         }
      },
      "OutputText":{
         "Output":{
            "Module":"DialogProcessing",
            "Port":"OutputText"
         },
         "Input":{
            "Module":"",
            "Port":"OutputText"
         }
      },
      "InputSpeech":{
         "Output":{
            "Module":"",
            "Port":"InputSpeech"
         },
         "Input":{
            "Module":"SpeechRecognition",
```

```
      "Port":"InputSpeech"
   }
},
"SpeechSegment":{
   "Output":{
      "Module":"SpeechSynthesis(Emotion)",
      "Port":"SpeechSegment"
   },
   "Input":{
      "Module":"LipsAnimation",
      "Port":" SpeechSegment"
   }
},
"OutputSpeech":{
   "Output":{
      "Module":"SpeechSynthesis(Emotion)",
      "Port":"OutputSpeech"
   },
   "Input":{
      "Module":"",
      "Port":"OuputSpeech"
   }
},
"InputVideo":{
   "Output":{
      "Module":"",
      "Port":"InputVideo"
   },
   "Input":{
      "Module":"VideoAnalysis",
      "Port":"InputVideo"
   }
},
"OutputVideo":{
   "Output":{
      "Module":"LipsAnimation",
      "Port":"OutputVideo"
   },
   "Input":{
      "Module":"",
      "Port":"OutputVideo"
   }
},
"RecognisedText":{
   "Output":{
      "Module":"SpeechRecognition",
      "Port":"RecognisedText"
   },
   "Input":{
      "Module":"LanguageUnderstanding",
      "Port":"RecognisedText"
   }
},
"Meaning(Text)":{
   "Output":{
      "Module":"LanguageUnderstanding",
      "Port":"Meaning(Text)"
   },
   "Input":{
      "Module":"DialogProcessing",
      "Port":"Meaning(Text)"
   }
},
"Meaning(Video)":{
   "Output":{
      "Module":"VideoAnalysis",
      "Port":"Meaning(Video)"
   },
   "Input":{
      "Module":"DialogProcessing",
      "Port":"Meaning(Video)"
   }
},
"Emotion(Text)":{
```

```
                  "Output":{
                     "Module":"LanguageUnderstanding",
                     "Port":"Emotion(Text)"
                  },
                  "Input":{
                     "Module":"EmotionFusion",
                     "Port":"Emotion(Text)"
                  }
               },
               "Emotion(Speech)":{
                  "Output":{
                     "Module":"SpeechRecognition",
                     "Port":"Emotion(Speech)"
                  },
                  "Input":{
                     "Module":"EmotionFusion",
                     "Port":"Emotion(Speech)"
                  }
               },
               "Emotion(Video)":{
                  "Output":{
                     "Module":"VideoAnalysis",
                     "Port":"Emotion(Video)"
                  },
                  "Input":{
                     "Module":"EmotionFusion",
                     "Port":"Emotion(Video)"
                  }
               },
               "FinalEmotion1":{
                  "Output":{
                     "Module":"EmotionFusion",
                     "Port":"FinalEmotion"
                  },
                  "Input":{
                     "Module":"DialogProcessing",
                     "Port":"FinalEmotion"
                  }
               },
               "FinalEmotion2":{
                  "Output":{
                     "Module":"DialogProcessing",
                     "Port":"FinalEmotion"
                  },
                  "Input":{
                     "Module":"LipsAnimation",
                     "Port":"FinalEmotion"
                  }
               },
               "TextWithEmotion":{
                  "Output":{
                     "Module":"DialogProcessing",
                     "Port":"TextWithEmotion"
                  },
                  "Input":{
                     "Module":"SpeechSynthesis(Emotion)",
                     "Port":"TextWithEmotion"
                  }
               }
            }
         ],

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

2 AIM metadata

2.1 SpeechRecognition


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"CWE",
         "Version":"1",
         "aName":"SpeechRecognition"
      },
      "Version":"*",
      "Profile":"",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "Emotion_t":"{byte emotionDegree; string<256 emotionName; string<256
emotionSetName}"
         }
      ],
      "Description":"This AIM implements speech recognition function that converts speech
of user utterance to text.",
      "Ports":[
         {
            "Name":"InputSpeech",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognizedText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Emotion(Speech)",
            "Direction":"OutputInput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

      ],

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


2.2 Video Analysis


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"CWE",
         "Version":"1",
         "aName":"VideoAnalysis"
```

```
      },
      "Version":"*",
      "Profile":"""",
      "Types":[
         {
            "Video_t":"uint24[]",
            "Meaning_t":"{Tagging_t POS_tagging; Tagging_t NE_tagging; Tagging_t
dependency_tagging; Tagging_t SRL_tagging}",
            "Emotion_t":"{byte emotionDegree; string<256 emotionName; string<256
emotionSetName}"
         }
      ],
      "Description":"This AIM implements video analysis for MPAI-MMC CWE Use Case.",
      "Ports":[
         {
            "Name":"InputVideo",
            "Direction":"InputOutput",
            "Record_Type":"Video_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Emotion(Video)",
            "Direction":"OutputInput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Meaning(Video)",
            "Direction":"OutputInput",
            "Record_Type":"Meaning_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.3 Language Understanding

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"CWE",
         "Version":"1",
         "aName":"LanguageUnderstanding "
      },
      "Version":"*",
      "Profile":"""",
      "Description":"This AIM implements Language Understanding for MPAI-MMC CWE Use Case.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Meaning_t":"{Tagging_t POS_tagging; Tagging_t NE_tagging; Tagging_t
dependency_tagging; Tagging_t SRL_tagging}",
            "Emotion_t":"{byte emotionDegree; string<256 emotionName; string<256
emotionSetName}"
         }
      ],
      "Ports":[
```

```
            {
               "Name":"InputText_1",
               "Direction":"InputOutput",
               "Record_Type":"Text_t",
               "Type":"Software",
               "Protocol":""
            },
            {
               "Name":"InputText_2",
               "Direction":"InputOutput",
               "Record_Type":"Text_t",
               "Type":"Software",
               "Protocol":""
            },
            {
               "Name":"Meaning(Video)",
               "Direction":"OutputInput",
               "Record_Type":"Meaning_t",
               "Type":"Software",
               "Protocol":""
            },
            {
               "Name":"Emotion(Video)",
               "Direction":"OutputInput",
               "Record_Type":"Video_t",
               "Type":"Software",
               "Protocol":""
            }
         ],
         "AIMs":[

         ],
         "Topology":[

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

2.4 Emotion Fusion

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"CWE",
         "Version":"1",
         "aName":"EmotionFusion"
      },
      "Version":"*",
      "Profile":"“”",
      "Description":"This AIM implements Emotion Fusion function.",
      "Types":[
         {
            "Emotion_t":"{byte emotionDegree; string<256 emotionName; string<256
emotionSetName}"
         }
      ],
      "Ports":[
         {
            "Name":"Emotion(Speech)",
            "Direction":"InputOutput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Emotion(Text)",
```

```
            "Direction":"InputOutput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Emotion(Video)",
            "Direction":"InputOutput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"FinalEmotion_1",
            "Direction":"OutputInput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.5 Dialog Processing

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"CWE",
         "Version":"1",
         "aName":"DialogProcessing"
      },
      "Version":"*",
      "Profile":"“”",
      "Description":"This AIM implements Dialog Processing function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Meaning_t":"{Tagging_t POS_tagging; Tagging_t NE_tagging; Tagging_t
dependency_tagging; Tagging_t SRL_tagging}",
            "Emotion_t":"{byte emotionDegree; string<256 emotionName; string<256
emotionSetName}",
            "TextWithEmotion_t":"{Text_t Text_Part; Emotion_t Emotion_Part}",
            "InputSelection_t":"{Enum Text | Enum Speech}"
         }
      ],
      "Ports":[
         {
            "Name":"Meaning(Text)",
            "Direction":"InputOutput",
            "Record_Type":"Meaning_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Meaning(Video)",
            "Direction":"InputOutput",
            "Record_Type":"Meaning_t",
            "Type":"Software",
            "Protocol":""
```

```
         },
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"InputSelection_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText1",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Text(Speech)",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"FinalEmotion_1",
            "Direction":"InputOutput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TextWithEmotion",
            "Direction":"OutputInput",
            "Record_Type":"TextWithEmotion_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"FinalEmotion_2",
            "Direction":"OutputInput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.6 Speech Synthesis

```
 {
    "AIM":{
       "Implementer_ID":"###",
       "Standard":{
          "Name":"MMC",
          "Use_Case":"CWE",
          "Version":"1",
          "aName":"SpeechSynthesis"
       },
       "Version":"*",
       "Profile":"“”",
       "Description":"This AIM implements Speech Synthesis function.",
       "Types":[
```

```
            {
               "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
               "Speech_t":"uint16[]",
               "TextWithEmotion_t":"{Text_t Text_Part, Emotion_t Emotion_Part}"
            }
         ],
         "Ports":[
            {
               "Name":"TextWithEmotion",
               "Direction":"InputOutput",
               "Record_Type":"TextWithEmotion_t",
               "Type":"Software",
               "Protocol":""
            },
            {
               "Name":"Output(Speech)",
               "Direction":"OutputInput",
               "Record_Type":"Speech_t",
               "Type":"Software",
               "Protocol":""
            },
            {
               "Name":"SpeechSegment",
               "Direction":"OutputInput",
               "Record_Type":"Speech_t",
               "Type":"Software",
               "Protocol":""
            }
         ],
         "AIMs":[

         ],
         "Topology":[

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

2.7 Lips Animation

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"CWE",
         "Version":"1",
         "aName":"LipsAnimation"
      },
      "Version":"*",
      "Profile":"“”",
      "Description":"This AIM implements Lips Animation function.",
      "Types":[
         {
            "Speech_t":"uint16[]",
            "Video_t":"uint24[]",
            "Emotion_t":"{byte emotionDegree; string<256 emotionName; string<256
emotionSetName}"
         }
      ],
      "Ports":[
         {
            "Name":"SpeechSegment",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
```

```
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"FinalEmotion_2",
            "Direction":"InputOutput",
            "Record_Type":"Emotion_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputVideo",
            "Direction":"OutputInput",
            "Record_Type":"Video_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

Annex 5 – AIW and AIM Metadata of MMC-MQA

1 AIW metadata for MQA


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MQA",
         "Version":"1",
         "aName":"_MAIN_"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIW implements MPAI-MMC Multimodal Question Answering Use
Case",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "Video_t":"uint24[]",
            "ObjectIdentifier_t":"{string objectImageLabel; float32 confidenceLevel}",
            "Intention_t":"{string<256 qtopic; string<256 qfocus; string<256 qLAT;
string<256 qSAT; string<256 qdomain}",
            "Tagging_t":"{string<256 set; string<256 result}",
            "Meaning_t":"{Tagging_t POS_tagging; Tagging_t NE_tagging; Tagging_t
dependency_tagging; Tagging_t SRL_tagging}",
            "Emotion_t":"{byte emotionDegree; string<256 emotionName; string<256
emotionSetName}",
            "SpeechFeatures_t":"{byte pitch; string<256 tone; string<256 intonation;
string<256 intensity; string<256 speed; Emotion_t emotion; float32[] NNspeechFeatures}",
            "InputSelection_t":"{Enum Text | Enum Speech}"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"InputSelection_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText1",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText2",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputVideo",
            "Direction":"InputOutput",
            "Record_Type":"Video_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText",
```

```
      "Direction":"OutputInput",
      "Record_Type":"Text_t",
      "Type":"Software",
      "Protocol":""
   },
   {
      "Name":" OutputSpeech",
      "Direction":"OutputInput",
      "Record_Type":"Speech_t",
      "Type":"Software",
      "Protocol":""
   }
],
"AIMs":[
   {
      "SpeechRecogniton":"@*:(S:(MMC:MQA:2:SpeechRecogniton)):*",
      "VideoAnalysis":"@*:(S:(MMC:MQA:2:VideoAnalysis)):*",
      "LanguageUnderstanding":"@*:(S:(MMC:MQA:2:LanguageUnderstanding)):*",
      "QuestionAnalysis":"@*:(S:(MMC:MQA:2:QuestionAnalysis)):*",
      "QuestionAnswering":"@*:(S:(MMC:MQA:2:QuestionAnswering)):*",
      "SpeechSynthesis":"@*:(S:(MMC:MQA:2:SpeechSynthesis)):*"
   }
],
"Topology":[
   {
      "InputText1":{
         "Output":{
            "Module":"",
            "Port":"InputText1"
         },
         "Input":{
            "Module":"LanguageUnderstanding",
            "Port":"InputText1"
         }
      },
      "InputText2":{
         "Output":{
            "Module":"",
            "Port":"InputText2"
         },
         "Input":{
            "Module":"QuestionAnswering",
            "Port":"InputText2"
         }
      },
      "RecognisedText1":{
         "Output":{
            "Module":"SpeechRecognition",
            "Port":"RecognisedText1"
         },
         "Input":{
            "Module":"LanguageUnderstanding",
            "Port":"RecognisedText1"
         }
      },
      "RecognisedText2":{
         "Output":{
            "Module":"SpeechRecognition",
            "Port":"RecognisedText2"
         },
         "Input":{
            "Module":"QuestionAnswering",
            "Port":"RecognisedText2"
         }
      },
      "ReplyText":{
         "Output":{
            "Module":"QuestionAnswering",
            "Port":"ReplyText"
         },
         "Input":{
            "Module":"SpeechSynthesis",
            "Port":"ReplyText"
         }
```

```
},
"OutputText":{
   "Output":{
      "Module":"QuestionAnswering",
      "Port":"OutputText"
   },
   "Input":{
      "Module":"",
      "Port":"OutputText"
   }
},
"InputSpeech":{
   "Output":{
      "Module":"",
      "Port":"InputSpeech"
   },
   "Input":{
      "Module":"SpeechRecognition",
      "Port":"InputSpeech"
   }
},
"OutputSpeech":{
   "Output":{
      "Module":"SpeechSynthesis",
      "Port":"OutputSpeech"
   },
   "Input":{
      "Module":"",
      "Port":"OutputSpeech"
   }
},
"InputVideo":{
   "Output":{
      "Module":"",
      "Port":"InputVideo"
   },
   "Input":{
      "Module":"VideoAnalysis",
      "Port":"InputVideo"
   }
},
"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":"LanuguageUnderstanding",
                  "Port":"ObjectIdentifier"
               }
            }
         }
      ],
      "Documentation":[
         {
            "Type":"tutorial",
            "URI":"https://mpai.community/standards/mpai-mmc/"
         }
      ]
   }
}
```


2 AIM metadata

2.1 SpeechRecognition


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MQA",
         "Version":"1",
         "aName":"SpeechRecognition"
      },
      "Version":"*",
      "Profile":"“”",
      "Description":"This AIM implements speech recognition function that converts speech
to text of user utterance.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSpeech",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognisedText1",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognisedText2",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

      ],
      "Implementations":[
         {
```

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


2.2 Video Analysis


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MQA",
         "Version":"1",
         "aName":"VideoAnalysis"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements video analysis.",
      "Types":[
         {
            "Video_t":"uint24[]",
            "ObjectIdentifier_t":"{string objectImageLabel; float32 confidenceLevel}"
         }
      ],
      "Ports":[
         {
            "Name":"InputVideo",
            "Direction":"InputOutput",
            "Record_Type":"Video_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"ObjectIdentifier",
            "Direction":"OutputInput",
            "Record_Type":"ObjectIdentifier_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.3 Language Understanding


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MQA",
         "Version":"1",
         "aName":"LanguageUnderstanding "
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements Language Understanding function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "ObjectIdentifier_t":"{string objectImageLabel; float32 confidenceLevel}",
            "Meaning_t":"{Tagging_t POS_tagging; Tagging_t NE_tagging; Tagging_t
dependency_tagging; Tagging_t SRL_tagging}"
         }
      ],
      "Ports":[
         {
            "Name":"RecognisedText1",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText1",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"ObjectIdentifier",
            "Direction":"InputOutput",
            "Record_Type":" ObjectIdentifier_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Meaning_1",
            "Direction":"OutputInput",
            "Record_Type":"Meaning_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Meaning_2",
            "Direction":"OutputInput",
            "Record_Type":"Meaning_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.4 Question Analysis

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MQA",
         "Version":"1",
         "aName":"QuestionAnalysis"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements Question Analysis function.",
      "Types":[
         {
            "Intention_t":"{string<256 qtopic; string<256 qfocus; string<256 qLAT; string<256
qSAT; string<256 qdomain}",
            "Meaning_t":"{Tagging_t POS_tagging; Tagging_t NE_tagging; Tagging_t
dependency_tagging; Tagging_t SRL_tagging}"
         }
      ],
      "Ports":[
         {
            "Name":"Meaning_2",
            "Direction":"InputOutput",
            "Record_Type":"Meaning_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"Intention",
            "Direction":"OutputInput",
            "Record_Type":"Intention_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.5 Question Answering

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MQA",
         "Version":"1",
         "aName":"QuestionAnswering"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements Question Answering function.",
```

```
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Meaning_t":"{Tagging_t POS_tagging; Tagging_t NE_tagging; Tagging_t
dependency_tagging; Tagging_t SRL_tagging}"
         }
      ],
      "Ports":[
         {
            "Name":"Meaning_1",
            "Direction":"InputOutput",
            "Record_Type":"Meaning_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText2",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognisedText2",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"ReplyText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.6 Speech Synthesis (Text)

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MQA",
         "Version":"1",
         "aName":"SpeechSynthesis2"
      },
      "Version":"*",
```

```
      "Profile":"",
      "Description":"This AIM implements Speech Synthesis function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]"
         }
      ],
      "Ports":[
         {
            "Name":"ReplyText",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputSpeech",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

Annex 6 – AIW and AIM Metadata of MMC-UST

1 AIW metadata for UST


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"UST",
         "Version":"1",
         "aName":"_MAIN_"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIW implements UST application of MPAI-MMC",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "InputSelection_t":"{Enum Text | Enum Speech}",
            "Language_t":"uint8[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"InputSelection_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RequestedLanguage",
            "Direction":"InputOutput",
            "Record_Type":"uint8[5] Language_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech1",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech2",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedSpeech",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
```

```
   }
],
"AIMs":[
   {
      "SpeechRecogniton":"@*:(S:(MMC:UST:2:SpeechRecogniton)):*",
      "Translation":"*:(S:(MMC:UST:2:Translation)):*",
      "SpeechFeatureExtraction":"@*:(S:(MMC:UST:2:LanguageUnderstanding)):*",
      "SpeechSynthesis":"@*:(S:(MMC:UST:2:SpeechSynthesis)):*"
   }
],
"Topology":[
   {
      "RequestedLanguage":{
         "Output":{
            "Module":"",
            "Port":"RequestedLanguage"
         },
         "Input":{
            "Module":"Translation",
            "Port":"RequestedLanguage"
         }
      },
      "InputText":{
         "Output":{
            "Module":"",
            "Port":"InputText"
         },
         "Input":{
            "Module":"Translation",
            "Port":"InputText "
         }
      },
      "InputSpeech1":{
         "Output":{
            "Module":"",
            "Port":"InputSpeech1"
         },
         "Input":{
            "Module":"SpeechRecognition",
            "Port":"InputSpeech1"
         }
      },
      "InputSpeech2":{
         "Output":{
            "Module":"",
            "Port":"InputSpeech2"
         },
         "Input":{
            "Module":"SpeechFeatureExtraction",
            "Port":"InputSpeech2"
         }
      },
      "OutputSpeech":{
         "Output":{
            "Module":"SpeechSynthesis",
            "Port":"TranslatedSpeech"
         },
         "Input":{
            "Module":"",
            "Port":"TranslatedSpeech"
         }
      },
      "SpeechFeatures":{
         "Output":{
            "Module":"SpeechFeatureExtraction",
            "Port":"SpeechFeatures"
         },
         "Input":{
            "Module":"SpeechSynthesis",
            "Port":"SpeechFeatures"
         }
      },
      "RecognizedText":{
         "Output":{
```

```
                  "Module":"SpeechRecognition",
                  "Port":"RecognizedText"
               },
               "Input":{
                  "Module":"Translation",
                  "Port":"RecognizedText"
               }
            },
            "TranslatedText":{
               "Output":{
                  "Module":"Translation",
                  "Port":"TranslatedText"
               },
               "Input":{
                  "Module":"SpeechSynthesis",
                  "Port":"TranslatedText"
               }
            },
            "OutputText":{
               "Output":{
                  "Module":"Translation",
                  "Port":"TranslatedText"
               },
               "Input":{
                  "Module":"",
                  "Port":"TranslatedText"
               }
            }
         }
      ],

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


2 AIM metadata

2.1 SpeechRecognition


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"UST",
         "Version":"1",
         "aName":"SpeechRecognition"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements speech recognition function for UST that converts
speech to text of user utterance.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSpeech1",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognizedText",
```

```
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

      ],

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

2.2 Translation

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"UST",
         "Version":"2",
         "aName":"Translation"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements Translation function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "InputSelection_t":"{Enum Text | Enum Speech}",
            "Language_t":"uint8[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"InputSelection_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RequestedLanguage",
            "Direction":"InputOutput",
            "Record_Type":"uint8[5] Language_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
```

```
         },
         {
            "Name":"TranslatedText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.3 Speech Feature Extraction

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"UST",
         "Version":"1",
         "aName":"SpeechFeatureExtraction "
      },
      "Version":"*",
      "Profile":" ",
      "Description":"This AIM implements Speech Feature Extraction function.",
      "Types":[
         {
            "Speech_t":"uint16[]",
            "SpeechFeatures_t":"{byte pitch; string<256 tone; string<256 intonation;
string<256 intensity; string<256 speed; Emotion_t emotion; float32[] NNspeechFeatures}"
         }
      ],
      "Ports":[
         {
            "Name":"InputSpeech2",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"SpeechFeatures",
            "Direction":"OutputInput",
            "Record_Type":"SpeechFeatures_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

```
         }
      ]
   }
}
```

2.4 Speech Synthesis

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"UST",
         "Version":"2",
         "aName":"SpeechSynthesis"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements Speech Synthesis function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "SpeechFeatures_t":"{byte pitch; string<256 tone; string<256 intonation;
string<256 intensity; string<256 speed; Emotion_t emotion; float32[] NNspeechFeatures}"
         }
      ],
      "Ports":[
         {
            "Name":"TranslatedText",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"SpeechFeatures",
            "Direction":"InputOutput",
            "Record_Type":"SpeechFeatures_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputSpeech",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

Annex 7 – AIW and AIM Metadata of MMC-BST

1 AIW metadata for BST


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"BST",
         "Version":"1",
         "aName":"_MAIN_"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIW implements BST application of MPAI-MMC",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "InputSelection_t":"{Enum Text | Enum Speech}",
            "Language_t":"uint8[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"InputSelection_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RequestedLanguage",
            "Direction":"InputOutput",
            "Record_Type":"uint8[5] Language_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText1",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText2",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech1",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech2",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech3",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
```

```
    },
    {
       "Name":"InputSpeech4",
       "Direction":"InputOutput",
       "Record_Type":"Speech_t",
       "Type":"Software",
       "Protocol":""
    },
    {
       "Name":"TranslatedText1",
       "Direction":"OutputInput",
       "Record_Type":"Text_t",
       "Type":"Software",
       "Protocol":""
    },
    {
       "Name":"TranslatedText2",
       "Direction":"OutputInput",
       "Record_Type":"Text_t",
       "Type":"Software",
       "Protocol":""
    },
    {
       "Name":"TranslatedSpeech1",
       "Direction":"OutputInput",
       "Record_Type":"Speech_t",
       "Type":"Software",
       "Protocol":""
    },
    {
       "Name":"TranslatedSpeech2",
       "Direction":"OutputInput",
       "Record_Type":"Speech_t",
       "Type":"Software",
       "Protocol":""
    }
],
"AIMs":[
    {
       "SpeechRecogniton":"@*:(S:(MMC:BST:2:SpeechRecogniton)):*",
       "Translation":"@*:(S:(MMC:BST:2:Translation)):*",
       "SpeechFeatureExtraction":"@*:(S:(MMC:BST:2:LanguageUnderstanding)):*",
       "SpeechSynthesis":"@*:(S:(MMC:BST:2:SpeechSynthesis)):*"
    }
],
"Topology":[
    {
       "RequestedLanguage":{
          "Output":{
             "Module":"",
             "Port":"RequestedLanguage "
          },
          "Input":{
             "Module":"Translation",
             "Port":"RequestedLanguage"
          }
       },
       "InputText1":{
          "Output":{
             "Module":"",
             "Port":"InputText1"
          },
          "Input":{
             "Module":"Translation",
             "Port":"InputText1 "
          }
       },
       "InputText2":{
          "Output":{
             "Module":"",
             "Port":"InputText2"
          },
          "Input":{
             "Module":"Translation",
```

```
      "Port":"InputText2"
   }
},
"InputSpeech1":{
   "Output":{
      "Module":"",
      "Port":"InputSpeech1"
   },
   "Input":{
      "Module":"SpeechRecognition",
      "Port":"InputSpeech1"
   }
},
"InputSpeech2":{
   "Output":{
      "Module":"",
      "Port":"InputSpeech2"
   },
   "Input":{
      "Module":"SpeechRecognition",
      "Port":"InputSpeech2"
   }
},
"InputSpeech3":{
   "Output":{
      "Module":"",
      "Port":"InputSpeech3"
   },
   "Input":{
      "Module":"SpeechFeatureExtraction",
      "Port":"InputSpeech3"
   }
},
"InputSpeech4":{
   "Output":{
      "Module":"",
      "Port":"InputSpeech4"
   },
   "Input":{
      "Module":"SpeechFeatureExtraction",
      "Port":"InputSpeech4"
   }
},
"TranslatedSpeech1":{
   "Output":{
      "Module":"SpeechSynthesis",
      "Port":"TranslatedSpeech1"
   },
   "Input":{
      "Module":"",
      "Port":"TranslatedSpeech1"
   }
},
"TranslatedSpeech2":{
   "Output":{
      "Module":"SpeechSynthesis",
      "Port":"TranslatedSpeech2"
   },
   "Input":{
      "Module":"",
      "Port":"TranslatedSpeech2"
   }
},
"SpeechFeatures1":{
   "Output":{
      "Module":"SpeechFeatureExtraction",
      "Port":"SpeechFeatures1"
   },
   "Input":{
      "Module":"SpeechSynthesis",
      "Port":"SpeechFeatures1"
   }
},
"SpeechFeatures2":{
```

```
            "Output":{
               "Module":"SpeechFeatureExtraction",
               "Port":"SpeechFeatures2"
            },
            "Input":{
               "Module":"SpeechSynthesis",
               "Port":"SpeechFeatures2"
            }
         },
         "RecognizedText1":{
            "Output":{
               "Module":"SpeechRecognition",
               "Port":"RecognizedText1"
            },
            "Input":{
               "Module":"Translation",
               "Port":"RecognizedText1"
            }
         },
         "RecognizedText2":{
            "Output":{
               "Module":"SpeechRecognition",
               "Port":"RecognizedText2"
            },
            "Input":{
               "Module":"Translation",
               "Port":"RecognizedText2"
            }
         },
         "TranslatedText1":{
            "Output":{
               "Module":"Translation",
               "Port":"TranslatedText1"
            },
            "Input":{
               "Module":"",
               "Port":"TranslatedText1"
            }
         },
         "TranslatedText2":{
            "Output":{
               "Module":"Translation",
               "Port":"TranslatedText2"
            },
            "Input":{
               "Module":"",
               "Port":"TranslatedText2"
            }
         },
         "TranslatedText3":{
            "Output":{
               "Module":"Translation",
               "Port":"TranslatedText3"
            },
            "Input":{
               "Module":"SpeechSynthesis",
               "Port":"TranslatedText3"
            }
         },
         "TranslatedText4":{
            "Output":{
               "Module":"Translation",
               "Port":"TranslatedText4"
            },
            "Input":{
               "Module":"SpeechSynthesis",
               "Port":"TranslatedText4"
            }
         }
      }
   ],
   "Documentation":[
      {
         "Type":"tutorial",
```

```
            "URI":"https://mpai.community/standards/mpai-mmc/"
         }
      ]
   }
}
```


2 AIM metadata

2.1 SpeechRecognition


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"BST",
         "Version":"2",
         "aName":"SpeechRecognition3"
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIM implements speech recognition function for BST that converts
speech to text of user utterance.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSpeech1",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech2",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognizedText1",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognizedText2",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.2 Translation


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"UST",
         "Version":"2",
         "aName":"Translation"
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIM implements Translation function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "InputSelection_t":"{Enum Text | Enum Speech}",
            "Language_t":"uint8[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"InputSelection_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RequestedLanguages",
            "Direction":"InputOutput",
            "Record_Type":"uint8[5] Language_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText1",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText2",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText1",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText2",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText3",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
```

```
            },
            {
               "Name":"TranslatedText4",
               "Direction":"OutputInput",
               "Record_Type":"Text_t",
               "Type":"Software",
               "Protocol":""
            }
         ],
         "AIMs":[

         ],
         "Topology":[

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

2.3 Speech Feature Extraction

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"BST",
         "Version":"1",
         "aName":"SpeechFeatureExtraction "
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIM implements Speech Feature Extraction function.",
      "Types":[
         {
            "Speech_t":"uint16[]",
            "SpeechFeatures_t":"{byte pitch; string<256 tone; string<256 intonation;
string<256 intensity; string<256 speed; Emotion_t emotion; float32[] NNspeechFeatures}"
         }
      ],
      "Ports":[
         {
            "Name":"InputSpeech3",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech4",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"SpeechFeatures1",
            "Direction":"OutputInput",
            "Record_Type":"SpeechFeatures_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"SpeechFeatures2",
            "Direction":"OutputInput",
            "Record_Type":"SpeechFeatures_t",
```

```
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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


2.4 Speech Synthesis


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"BST",
         "Version":"1",
         "aName":"SpeechSynthesis"
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIM implements Speech Synthesis function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "SpeechFeatures_t":"{byte pitch; string<256 tone; string<256 intonation;
string<256 intensity; string<256 speed; Emotion_t emotion; float32[] NNspeechFeatures}"
         }
      ],
      "Ports":[
         {
            "Name":"TranslatedText3",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText4",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"SpeechFeatures1",
            "Direction":"InputOutput",
            "Record_Type":"SpeechFeatures_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"SpeechFeatures2",
            "Direction":"InputOutput",
            "Record_Type":"SpeechFeatures_t",
            "Type":"Software",
            "Protocol":""
         },
         {
```

```
            "Name":"TranslatedSpeech1",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedSpeech2",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

Annex 8 – AIW and AIM Metadata of MMC-MST

1 AIW metadata for MST


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MST",
         "Version":"1",
         "aName":"_MAIN_"
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIW implements MST application of MPAI-MMC",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "Speech_t":"uint16[]",
            "InputSelection_t":"{Enum Text | Enum Speech}",
            "Language_t":"uint8[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"{Enum Text | Enum Speech}",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RequestedLanguage",
            "Direction":"InputOutput",
            "Record_Type":"uint8[5] Language_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputText",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech1",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InputSpeech2",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText1",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText2",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
```

```
    },
    {
       "Name":"OutputTextN",
       "Direction":"OutputInput",
       "Record_Type":"Text_t",
       "Type":"Software",
       "Protocol":""
    },
    {
       "Name":"InterpretedSpeech1",
       "Direction":"OutputInput",
       "Record_Type":"Speech_t",
       "Type":"Software",
       "Protocol":""
    },
    {
       "Name":"InterpretedSpeech2",
       "Direction":"OutputInput",
       "Record_Type":"Speech_t",
       "Type":"Software",
       "Protocol":""
    },
    {
       "Name":"InterpretedSpeechN",
       "Direction":"OutputInput",
       "Record_Type":"Speech_t",
       "Type":"Software",
       "Protocol":""
    }
],
"AIMs":[
    {
       "SpeechRecogniton":"@*:(S:(MMC:MST:2:SpeechRecogniton)):* ",
       "Translation":"@*:(S:(MMC:MST:2:Translation)):12",
       "SpeechFeatureExtraction":"@*:(S:(MMC:MST:2:LanguageUnderstanding)):* ",
       "SpeechSynthesis":"@*:(S:(MMC:MST:2:SpeechSynthesis)):* "
    }
],
"Topology":[
    {
       "RequestedLanguage":{
          "Output":{
             "Module":"",
             "Port":"RequestedLanguage"
          },
          "Input":{
             "Module":"Translation",
             "Port":"RequestedLanguage"
          }
       },
       "InputText":{
          "Output":{
             "Module":"",
             "Port":"InputText"
          },
          "Input":{
             "Module":"Translation",
             "Port":"InputText "
          }
       },
       "InputSpeech1":{
          "Output":{
             "Module":"",
             "Port":"InputSpeech1"
          },
          "Input":{
             "Module":"SpeechRecognition",
             "Port":"InputSpeech1"
          }
       },
       "InputSpeech2":{
          "Output":{
             "Module":"",
             "Port":"InputSpeech2"
```

```
   },
   "Input":{
      "Module":"SpeechFeatureExtraction",
      "Port":"InputSpeech2"
   }
},
"InterpretedSpeech1":{
   "Output":{
      "Module":"SpeechSynthesis",
      "Port":"InterpretedSpeech1"
   },
   "Input":{
      "Module":"",
      "Port":"InterpretedSpeech1"
   }
},
"InterpretedSpeech2":{
   "Output":{
      "Module":"SpeechSynthesis",
      "Port":"InterpretedSpeech2"
   },
   "Input":{
      "Module":"",
      "Port":"InterpretedSpeech2"
   }
},
"InterpretedSpeechN":{
   "Output":{
      "Module":"SpeechSynthesis",
      "Port":"InterpretedSpeechN"
   },
   "Input":{
      "Module":"",
      "Port":"InterpretedSpeechN"
   }
},
"SpeechFeatures":{
   "Output":{
      "Module":"SpeechFeatureExtraction",
      "Port":"SpeechFeatures"
   },
   "Input":{
      "Module":"SpeechSynthesis",
      "Port":"SpeechFeatures"
   }
},
"RecognizedText":{
   "Output":{
      "Module":"SpeechRecognition",
      "Port":"RecognizedText"
   },
   "Input":{
      "Module":"Translation",
      "Port":"RecognizedText"
   }
},
"TranslatedText1":{
   "Output":{
      "Module":"Translation",
      "Port":"TranslatedText1"
   },
   "Input":{
      "Module":"SpeechSynthesis",
      "Port":"TranslatedText1"
   }
},
"TranslatedText2":{
   "Output":{
      "Module":"Translation",
      "Port":"TranslatedText2"
   },
   "Input":{
      "Module":"SpeechSynthesis",
      "Port":"TranslatedText2"
```

```
            }
         },
         "TranslatedTextN":{
            "Output":{
               "Module":"Translation",
               "Port":"TranslatedTextN"
            },
            "Input":{
               "Module":"SpeechSynthesis",
               "Port":"TranslatedTextN"
            }
         },
         "OutputText1":{
            "Output":{
               "Module":"Translation",
               "Port":"OutputText1"
            },
            "Input":{
               "Module":"",
               "Port":"OutputText1"
            }
         },
         "OutputText2":{
            "Output":{
               "Module":"Translation",
               "Port":"OutputText2"
            },
            "Input":{
               "Module":"",
               "Port":"OutputText2"
            }
         },
         "OutputTextN":{
            "Output":{
               "Module":"Translation",
               "Port":"OutputTextN"
            },
            "Input":{
               "Module":"",
               "Port":"OutputTextN"
            }
         }
      }
   ],
   "Documentation":[
      {
         "Type":"tutorial",
         "URI":"https://mpai.community/standards/mpai-mmc/"
      }
   ]
  }
}
```


2 AIM metadata

2.1 SpeechRecognition


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MST",
         "Version":"1",
         "aName":"SpeechRecognition"
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIM implements speech recognition function for MST that converts
speech to text of user utterance.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
```

```
            "Speech_t":"uint16[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSpeech1",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RecognizedText",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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


2.2 Translation


```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MST",
         "Version":"1",
         "aName":"Translation"
      },
      "Version":"*",
      "Profile":"",
      "Description":"This AIM implements Translation function.",
      "Types":[
         {
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "InputSelection_t":"{Enum Text | Enum Speech}",
            "Language_t":"uint8[]"
         }
      ],
      "Ports":[
         {
            "Name":"InputSelection",
            "Direction":"InputOutput",
            "Record_Type":"InputSelection_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"RequestedLanguage",
            "Direction":"InputOutput",
            "Record_Type":"uint8[5] Language_t",
            "Type":"Software",
            "Protocol":""
         },
         {
```

```
            "Name":"InputText",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText1",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText2",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedTextN",
            "Direction":"OutputInput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.3 Speech Feature Extraction

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"MST",
         "Version":"1",
         "aName":"SpeechFeatureExtraction "
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIM implements Speech Feature Extraction function for MST.",
      "Types":[
         {
            "Speech_t":"uint16[]",
            "SpeechFeatures_t":"{byte pitch; string<256 tone; string<256 intonation;
string<256 intensity; string<256 speed; Emotion_t emotion; float32[] NNspeechFeatures}"
         }
      ],
      "Ports":[
         {
            "Name":"InputSpeech2",
            "Direction":"InputOutput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
```

```
         },
         {
            "Name":"SpeechFeatures",
            "Direction":"OutputInput",
            "Record_Type":"SpeechFeatures_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

2.4 Speech Synthesis

```
{
   "AIM":{
      "Implementer_ID":"###",
      "Standard":{
         "Name":"MMC",
         "Use_Case":"OMT",
         "Version":"2",
         "aName":"SpeechSynthesis"
      },
      "Version":"1",
      "Profile":"",
      "Description":"This AIM implements Speech Synthesis function for OMT.",
      "Types":[
         {
            "Speech_t":"uint16[]",
            "Text_t":"{byte[] One_Byte_Text | uint16[] Two_Byte_Text}",
            "SpeechFeatures_t":"{byte pitch; string<256 tone; string<256 intonation;
string<256 intensity; string<256 speed; Emotion_t emotion; float32[] NNspeechFeatures}"
         }
      ],
      "Ports":[
         {
            "Name":"TranslatedText1",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedText2",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"TranslatedTextN",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText1",
            "Direction":"InputOutput",
```

```
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputText2",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"OutputTextN",
            "Direction":"InputOutput",
            "Record_Type":"Text_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"SpeechFeatures",
            "Direction":"InputOutput",
            "Record_Type":"SpeechFeatures_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InterpretedSpeech1",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InterpretedSpeech2",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         },
         {
            "Name":"InterpretedSpeechN",
            "Direction":"OutputInput",
            "Record_Type":"Speech_t",
            "Type":"Software",
            "Protocol":""
         }
      ],
      "AIMs":[

      ],
      "Topology":[

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

Annex 9 – Communication Among AIM Implementors

A core design principle of MPAI is modularity: AI Modules or AIMs and their interfaces must be defined such that each AIM can be built by an independent implementor, without damage to the function of a use case as a whole.

However, MPAI also recognizes that that AIMs and their implementors may sometimes profit from communication and interchange of data and/or components. Such exchanges can be especially appropriate for AIMs featuring neural network components or comparable elements for machine learning – an increasingly common and important situation in the design of cooperative artificial intelligence modules.

On example is offed by the Emotionally Enhanced Speech (EES) use case, which is designed to allow addition of emotional features to an Emotionless Speech input element. In that use case, a Speech Feature Analyser2 AIM can issue queries to an Emotion KB AIM in order to fetch speech features that can convey named emotions, like “angry” or “sad,” belonging to a predefined Emotion Set, and can pass these to an Emotion Inserter AIM. However, while the three AIMs can indeed be independently implemented, Emotion Inserter must ultimately be able to process the returned speech features appropriately. If the latter incorporates neural networks, then for processing purposes it will need either (1) a usable neural model whose training has included features like those that Emotion KB will return or (2) a precise specification of the features themselves plus an adequate training corpus, so that Emotion Inserter can create its own usable model.

Comparable considerations obtain for the Conversation with Emotion (CWE) use case. And, more generally, they will obtain for any AIMs that exchange neural information. In explicitly providing for such communication among artificial machine learning models and components, MPAI is not only recognizing practical requirements for cooperation among such modules, but also acknowledging an analogy with communication among biological neural subsystems.