INTERNATIONAL STANDARD

ISO/IEC 23093-3

Second edition
2022-06

Information technology — Internet of media things —

Part 3: Media data formats and APIs

Technologies de l'information — Internet des objets media — Partie 3: API et formats des données

Reference number
ISO/IEC 23093-3:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 23093-3:2019), which has been technically revised.

The main changes are as follows:

— Addition of APIs for new MSensors, MActuators, and MAnalysers;

— Addition of data types for new MSensors, MActuators, and MAnalysers;

— Provide APIs to describe MPEG-V sensors and actuators;

— Addition of binary representation and its semantics.

A list of all parts in the ISO/IEC 23093 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

The ISO/IEC 23093 series provides an architecture and specifies APIs and compressed representation of data flowing between media things.

The APIs for the media things facilitate discovering other media things in the network, connecting and efficiently exchanging data between media things. The APIs also support transaction tokens to access valuable functionalities, resources, and data from media things.

Media things related information consists of characteristics and discovery data, setup information from a system designer, raw and processed sensed data, and actuation information. The ISO/IEC 23093 series specifies input and output data formats for media sensors, media actuators, media storages, media analysers, etc. In addition, media analysers can process sensed data from media sensors to produce analysed data, and the media analysers can be cascaded to extract semantic information.

This document contains the tools to describe data exchanged between media things (e.g., media sensors, media actuators, media analysers, media storages) and their APIs. It addresses the normative aspects of the data and APIs for media things and also illustrates non-normative examples.

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of patents.

ISO and the IEC take no position concerning the evidence, validity, and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patents right are registered with ISO and IEC. Information may be obtained from the patent database available at www.iso.org/patents.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified in the patent database. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Information technology — Internet of media things —

Part 3:
Media data formats and APIs

1 Scope

This document specifies the syntax and semantics of description schemes to represent data exchanged by media things (e.g., media sensors, media actuators, media analysers, media storages). Moreover, it specifies the APIs to exchange these data between media things.

This document does not specify how sensing and analysing is carried out but defines the interfaces between the media things.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 15938-3, *Information technology — Multimedia content description interface — Part 3: Visual*

ISO/IEC 15938-5, *Information technology — Multimedia content description interface — Part 5: Multimedia description schemes*

ISO/IEC 23005-2, *Information technology — Media context and control — Part 2: Control information*

ISO/IEC 23005-5, *Information technology — Media context and control — Part 5: Data formats for interaction devices*

ISO/IEC 23093-1, *Information technology — Internet of media things — Part 1: Architecture*

ISO/IEC 23093-2, *Information technology — Internet of media things — Part 2: Discovery and communication API*

3 Terms, definitions, and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 23093-1 and ISO/IEC 23093-2 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1.1
media actuator
MActuator
MThing that can actuate

3.1.2
media aggregator
MAggregator
MThing that contains multiple MThings

3.1.3
media analyser
MAnalyser
MThing that can analyse media or metadata and produce interpreted media, metadata, or commands

3.1.4
media manager
MManager
MThing that can register a list of MThings or be facilitated to search other MThings

3.1.5
media sensor
MSensor
MThing that can sense and produce media data

3.1.6
media storage
MStorage
MThing that can save media or metadata

3.2 Abbreviated terms

API application programming interface

MACV media actuator command vocabulary

MAOV media analyser output vocabulary

MSOV media sensor output vocabulary

MTDL media thing description language

SCDV sensor capability description vocabulary

XML extensible mark-up language

XSI XML streaming instructions

3.3 Schema documents

In the main text of this document, the syntax and semantics of data interfacing MThings are provided whenever possible as a single schema document.

In some cases, though, particularly for Clauses 6, 7, and 8, the syntax of data interfacing MThings is provided as a collection of schema snippets imbricated with other text. To form a valid schema document, users can gather these schema components in the same document with the schema wrapper provided at the head of the clause. For better readability, the relevant schema documents are supplied in https://standards.iso.org/iso-iec/23093/-3/ed-2/en/.

In all cases, each schema document has a `version` attribute, the value of which is "ISO/IEC 23093-3." Furthermore, an informative identifier is given as the value of the `id` attribute of the `schema` component. This identifier is non-normative and used as a convention in this document to reference another schema document. In particular, it is used for the `schemaLocation` attribute of the `include` and `import` schema components.

3.4 Use of prefixes

For clarity, throughout this document, consistent namespace prefixes are used.

"`xsi:`" prefix is not normative. It is a naming convention in this document to refer to an element of the `http://www.w3.org/2001/XMLSchema-instance` namespace.

"`xml:`" and "`xmlns:`" are normative prefixes defined in Reference [1]. The prefix "`xml:`" is by definition bound to "`http://www.w3.org/XML/1998/namespace`." The prefix "`xmlns:`" is used only for namespace bindings and is not itself bound to any namespace name.

All other prefixes used in either the text or examples of this document are not normative, e.g., "`mtdl:`", "`msov:`", "`macv:`", "`maov:`", "`mpeg7:`", "`scdv:`".

In particular, most informative examples in this document are provided as XML fragments without the typically required XML document declaration and, thus, miss a correct namespace binding context declaration. In these descriptions fragments, the different prefixes are bound to the `namespaces` as given in Table 1.

Table 1 — Mapping of prefixes to namespaces in examples and text

Prefix	Corresponding namespace
`scdv`	`urn:mpeg:mpeg-v:2017:01-SCDV-NS`
`mpeg7`	`urn:mpeg:mpeg7:schema:2004`
`mtdl`	`urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS`
`msov`	`urn:mpeg:mpeg-IoMT:2021:01-MSOV-NS`
`macv`	`urn:mpeg:mpeg-IoMT:2021:01-MACV-NS`
`maov`	`urn:mpeg:mpeg-IoMT:2021:01-MAOV-NS`
`xsi`	`http://www.w3.org/2001/XMLSchema-instance`
`xsd`	`http://www.w3.org/2001/XMLSchema`

Unlike the informative descriptions examples, the normative specification of the syntax of tools in XML schema follows the namespace binding context defined in the relevant schema declaration, such as the one described in 6.2.

4 APIs

4.1 General

This subclause specifies APIs and their descriptions to operate MThings and exchange structured data between MThings. Figure 1 shows an example of "GET" and "SET" functions invoked between MThings. For example, an MSensor should have "GET" functions to evoke and provide its sensed data. An MStorage should have "SET" functions to save sensed data obtained by an MSensor or to save analysed data provided by a MAnalyser. A MAnalyser should provide "GET" functions to produce metadata by analysing sensed data from MSensors or by generating metadata by analysing data fed by other MAnalysers. Finally, a MActuator should provide "SET" functions to control its functionalities. If there is no structured data exchanged between MThings, each MThing can have simple "SET" functions controlled by other MThings.

Figure 2 demonstrates an example of a function call sequence diagram between MThings. A face verifier (AS2) requests detected face regions extracted from the image (i.e., sensed data from S1) to the face region detector (AS1) by invoking the function `getFaceRegions()`. After receiving the function call, a face region detector (AS1) requests an image to a camera (S1) by invoking the function `getImageURL()`. The camera (S1) sends back the corresponding URL to the face region detector (AS1). In this case, the return type of the URL is a simple string. Suppose an MSensor returns data with standard structures. In that case, the data can be delivered by the return type class, either "MPEGVSensedDataType" or "SensedDataType," which can be described by XML or Binary representation.

After detecting the face region from the input image, the face region detector (AS1) sends back the recognised face regions with the standard structure to the face verifier (AS2). The data provided by a MAnalyser can be delivered by either a simple string like a URL or the return type class called "AnalysedDataType," which can be described by XML or Binary representation.

Finally, the face verifier (AS2) invokes the function `setLight()` to actuate (e.g., generate the coloured light) the lighting device (AC1). Again, the actuation data feeding to a MActuator can be delivered by a simple string like a URL or the return type class of "MPEGVCommandType" or "ActuationDataType," which can be described by XML or Binary representation.

The function calls trigger MThings either to generate and exchange data or to control MThings.

The function definitions (APIs) are defined for MSensor, MActuator, MAnalyser, MStorage, MManager, MAggregator, and return type classes in the following subclauses.

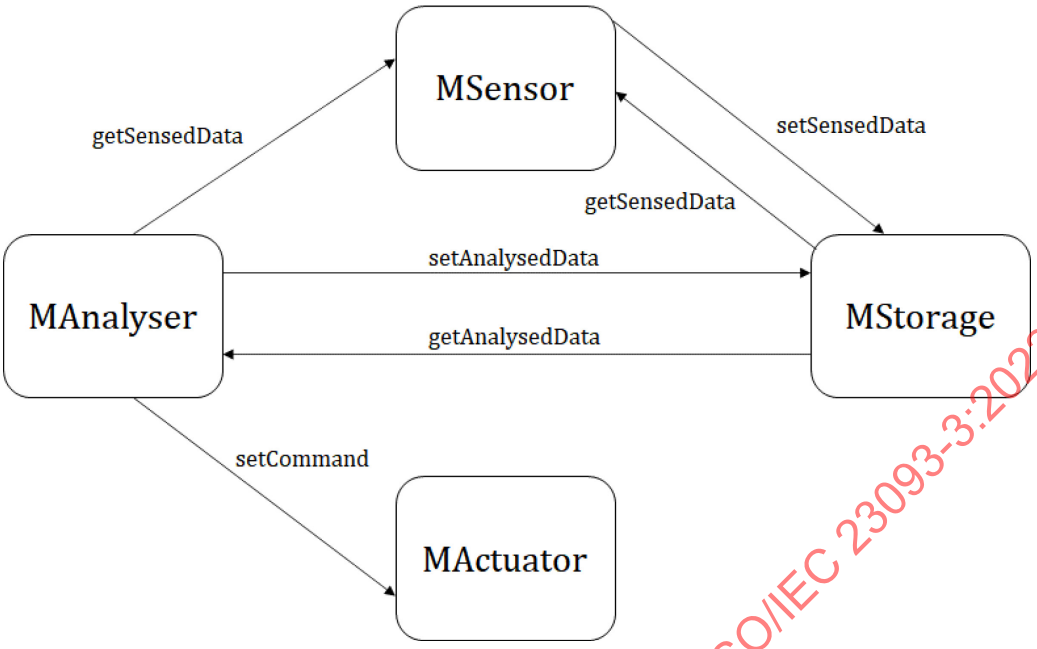


Figure 1 — Function invocation between MThings

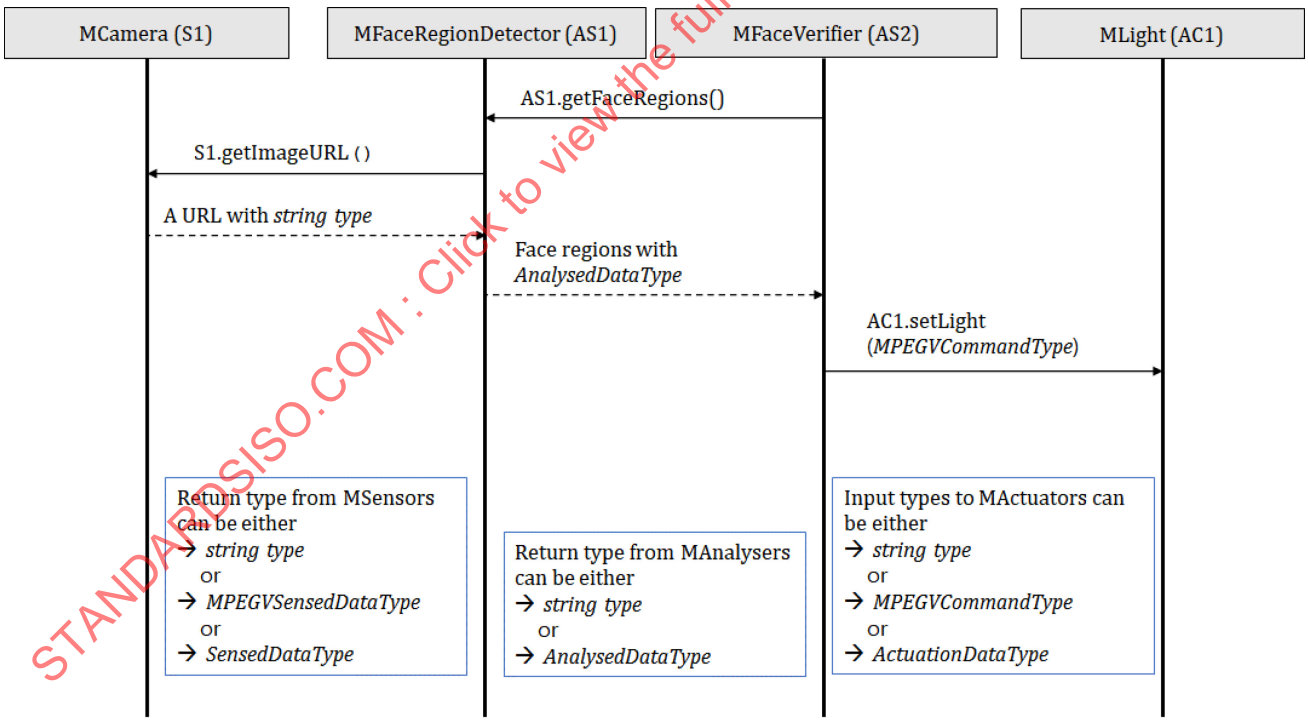


Figure 2 — Sequence diagram of function calls between MThings

4.2 APIs for IoMT sensors

4.2.1 General

This subclause defines API classes of IoMT sensors.

4.2.2 MSensor class

4.2.2.1 General

This subclause defines an MSensor class that shall inherit the features of MThing class defined in ISO/IEC 23093-2.

4.2.2.2 APIs

Table 2 presents the basic APIs of MSensor.

Table 2 — MSensor APIs

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSensor()	
Default constructor.	
MSensor(string id)	
MSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVCapabilityType	getMPEGVCapability()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensor capabilities from ISO/IEC 23005-2 (MPEG-V Part 2).
MPEGVSensedDataType	getMPEGVSensedData()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data from ISO/IEC 23005-5 (MPEG-V Part 5).

CapabilityListType	getSensorCapabilityList();
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a capability list specified in A.2.1 (Table A.2).
CapabilityListType	getAvailableSensorCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an available capability list specified in A.2.1 (Table A.2).
CapabilityListType	getAppliedSensorCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an applied capability list specified in A.2.1 (Table A.2).
SensedDataType	getCapturedTime()
	This function returns a captured time of sensed data.
SensedDataType	getCapturedTime(string tid)
	This function returns a captured time of sensed data. The `tid` is the transaction ID of payment for using this function.
float	getCapturedTime_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getCapturedTime()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5 (Table A.21, A.22). If the requested token is not supported, returns −1. Ex) getCapturedTime_Cost(0, "BTC") or getCapturedTime_Cost(0, "00000001") Ex) getCapturedTime_Cost(1, "USD") or getCapturedTime_Cost(1, "10010100")

4.2.3 API for IoMT microphone

4.2.3.1 General

This subclause defines a class of an IoMT microphone that shall inherit the features of `MSensor` class. The main functions of an IoMT microphone are composed of sending an audio URL, an audio sampling rate, microphone sensed data and its type, which is acquired by the microphone to other MThings.

4.2.3.2 APIs

Table 3 presents the APIs of an IoMT microphone.

Table 3 — IoMT microphone API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MMicrophone()	
Default constructor.	
MMicrophone(string id)	
MMicrophone(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVMicrophone()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MicrophoneSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVMicrophone(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MicrophoneSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVMicrophone_CostPerMinute(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens)per minute to use `getMPEGVMicrophone()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5 (Table A.21, A.22). If the requested token is not supported, returns -1. Ex) getMPEGVMicrophone_CostPerMinute(0, "BTC") or getMPEGVMicrophone_CostPerMinute(0, "00000001") Ex) getMPEGVMicrophone_CostPerMinute(1, "USD") or getMPEGVMicrophone_CostPerMinute(1, "10010100")
string	getAudioURL()
	This function returns a URL of an audio source.
string	getAudioURL(string tid)
	This function returns a URL of an audio source. The `tid` is the transaction ID of payment for using this function.
float	getAudioURL_CostPerMinute(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens)per minute to use `getAudioURL()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getAudioURL_CostPerMinute(0, "BTC") or getAudioURL_CostPerMinute(0, "00000001") Ex) getAudioURL_CostPerMinute(1, "USD") or getAudioURL_CostPerMinute(1, "10010100")
int	getAudioSamplingRate()
	This function returns a sampling rate of an audio source.
int	getAudioSamplingRate(string tid)
	This function returns a sampling rate of an audio source. The `tid` is the transaction ID of payment for using this function.

float	getAudioSamplingRate_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getAudioSamplingRate()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getAudioSamplingRate_Cost(0, "BTC") or getAudioSamplingRate_Cost(0, "00000001") Ex) getAudioSamplingRate_Cost(1, "USD") or getAudioSamplingRate_Cost(1, "10010100")

4.2.4 API for IoMT camera

4.2.4.1 General

This subclause defines a class of an IoMT camera that shall inherit the features of `MSensor` class. The main functions of an IoMT camera are sending a video URL, an image URL, camera sensed data and its type, which is acquired by the camera to other MThings.

4.2.4.2 APIs

Table 4 presents the APIs of an IoMT camera.

Table 4 — IoMT camera API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MCamera()	
Default constructor.	
MCamera(string id)	
MCamera(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVCamera()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `CameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVCamera(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `CameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVCamera_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) per minute to use `getMPEGVCamera()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVCamera_Cost(0, "BTC") or getMPEGVCamera_Cost(0, "00000001") Ex) getMPEGVCamera_Cost(1, "USD") or getMPEGVCamera_Cost(1, "10010100")
string	getVideoURL()
	This function returns a URL of a video source.
string	getVideoURL(string tid)
	This function returns a URL of a video source. The `tid` is the transaction ID of payment for using this function.

float	getVideoURL_CostPerMinute(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) per minute to use `getVideoURL()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getVideoURL_CostPerMinute(0, "BTC") or getVideoURL_CostPerMinute(0, "00000001") Ex) getVideoURL_CostPerMinute(1, "USD") or getVideoURL_CostPerMinute(1, "10010100")
string	getImageURL()
	This function returns a URL of an image source.
string	getImageURL(string tid)
	This function returns a URL of an image source. The `tid` is a token transaction ID of the corresponding service
float	getImageURL_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getImageURL()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getImageURL_Cost(0, "BTC") or getImageURL_Cost(0, "00000001") Ex) getImageURL_Cost(1, "USD") or getImageURL_Cost(1, "10010100")

4.2.5 API for IoMT RFID reader

4.2.5.1 General

This subclause defines a class of an IoMT RFID reader that shall inherit the features of `MSensor` class. The main functions of an IoMT RFID reader are composed of sending an RFID tag information sensed by the RFID reader to other MThings.

4.2.5.2 APIs

Table 5 presents the APIs of an IoMT RFID reader.

Table 5 — IoMT RFID reader API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MRFIDSensor()	
Default constructor.	
MRFIDSensor(string id)	
MRFIDSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
string	getRFIDtagInfo()
	This function returns RFID tag information.
string	getRFIDtagInfo(string tid)
	This function returns RFID tag information. The `tid` is the transaction ID of payment for using this function.
float	getRFIDtagInfo_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getRFIDtagInfo()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getRFIDtagInfo_Cost(0, "BTC") or getRFIDtagInfo_Cost(0, "00000001") Ex) getRFIDtagInfo_Cost(1, "USD") or getRFIDtagInfo_Cost(1, "10010100")

4.2.6 API for IoMT compass sensor

4.2.6.1 General

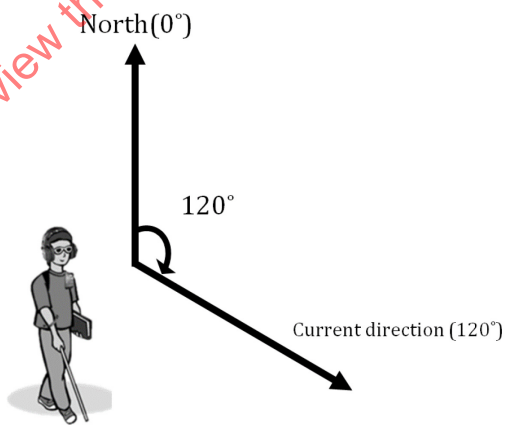
This subclause defines a class of an IoMT compass sensor that shall inherit the features of `MSensor` class. The main functions of an IoMT compass sensor are composed of sending an azimuth angle, geomagnetic sensed data, and its type, which is acquired by the compass sensor to other MThings.

4.2.6.2 APIs

Table 6 presents the APIs of an IoMT compass sensor.

Table 6 — IoMT compass sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MCompass()	
Default constructor.	
MCompass(string id)	
MCompass(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVGeomagnetic()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GeomagneticSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).

MPEGVSensedDataType	getMPEGVGeomagnetic(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GeomagneticSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVGeomagnetic_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVGeomagnetic()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVGeomagnetic_Cost(0, "BTC") or getMPEGVGeomagnetic_Cost(0, "00000001") Ex) getMPEGVGeomagnetic_Cost(1, "USD") or getMPEGVGeomagnetic_Cost(1, "10010100")
float	getAzimuthAngle()
	The function returns the current clockwise azimuth angle from the north (Figure 3).  **Figure 3 — Azimuth angle**
float	getAzimuthAngle(string tid)
	The function returns the current clockwise azimuth angle from the north. The `tid` is the transaction ID of payment for using this function.

float	getAzimuthAngle_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getAzimuthAngle()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getAzimuthAngle_Cost(0, "BTC") or getAzimuthAngle_Cost(0, "00000001") Ex) getAzimuthAngle_Cost(1, "USD") or getAzimuthAngle_Cost(1, "10010100")

4.2.7 API for IoMT orientation sensor

4.2.7.1 General

This subclause defines a class of an IoMT orientation sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT orientation sensor are composed of sending orientation sensed data and its type, which is acquired by the orientation sensor to other MThings.

4.2.7.2 APIs

Table 7 presents the APIs of an IoMT orientation sensor.

Table 7 — IoMT orientation sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MOrientationSensor()	
Default constructor.	
MOrientationSensor(string id)	
MOrientationSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVOrientation()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `OrientationSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVOrientation(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `OrientationSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVOrientation_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVOrientation()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVOrientation_Cost(0, "BTC") or getMPEGVOrientation_Cost(0, "00000001") Ex) getMPEGVOrientation_Cost(1, "USD") or getMPEGVOrientation_Cost(1, "10010100")

4.2.8 API for IoMT position sensor

4.2.8.1 General

This subclause defines a class of an IoMT position sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT position sensor are composed of sending positional sensed data and its type, which is acquired by the position sensor to other MThings.

4.2.8.2 APIs

Table 8 presents the APIs of an IoMT position sensor.

Table 8 — IoMT position sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MPositionSensor()	
Default constructor.	
MPositionSensor(string id)	
MPositionSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVPosition()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `PositionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVPosition(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `PositionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVPosition_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVPosition()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVPosition_Cost(0, "BTC") or getMPEGVPosition_Cost(0, "00000001") Ex) getMPEGVPosition_Cost(1, "USD") or getMPEGVPosition_Cost(1, "10010100")

4.2.9 API for IoMT global positioning sensor

4.2.9.1 General

This subclause defines a class of an IoMT global positioning sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT global positioning sensor are composed of sending latitude, longitude, global position sensed data, and type sensed by the global positioning sensor to other MThings.

4.2.9.2 APIs

Table 9 presents the APIs of an IoMT global positioning sensor.

Table 9 — IoMT global positioning sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MGlobalPositioningSensor()	
Default constructor.	
MGlobalPositioningSensor(string id)	
MGlobalPositioningSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
float	getLatitude()
	This function returns the value of latitude.
float	getLatitude(string tid)
	This function returns the value of latitude. The `tid` is the transaction ID of payment for using this function.
float	getLatitude_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getLatitude()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getLatitude_Cost(0, "BTC") or getLatitude_Cost(0, "00000001") Ex) getLatitude_Cost(1, "USD") or getLatitude_Cost(1, "10010100")
float	getLongitude()
	This function returns the value of longitude.
float	getLongitude(string tid)
	This function returns the value of longitude. The `tid` is the transaction ID of payment for using this function.
float	getLongitude_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getLongitude()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getLongitude_Cost(0, "BTC") or getLongitude_Cost(0, "00000001") Ex) getLongitude_Cost(1, "USD") or getLongitude_Cost(1, "10010100")

MPEGVSensedDataType	getMPEGVGlobalPosition()
	This method returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GlobalPositionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVGlobalPosition(string tid)
	This method returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GlobalPositionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVGlobalPosition_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVGlobalPosition()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVGlobalPosition_Cost(0, "BTC") or getMPEGVGlobalPosition_Cost(0, "00000001") Ex) getMPEGVGlobalPosition_Cost(1, "USD") or getMPEGVGlobalPosition_Cost(1, "10010100")

4.2.10 API for IoMT distance sensor

4.2.10.1 General

This subclause defines a class of an IoMT distance sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT distance sensor are composed of sending distance sensed data and its type, which is acquired by the distance sensor to other MThings.

4.2.10.2 APIs

Table 10 presents the APIs of an IoMT distance sensor.

Table 10 — IoMT distance sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MDistanceSensor()	

Default constructor.	
MDistanceSensor(string id)	
MDistanceSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVDistance()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DistanceSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVDistance(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DistanceSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVDistance_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVDistance()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVDistance_Cost(0, "BTC") or getMPEGVDistance_Cost(0, "00000001") Ex) getMPEGVDistance_Cost(1, "USD") or getMPEGVDistance_Cost(1, "10010100")

4.2.11 API for IoMT light sensor

4.2.11.1 General

This subclause defines a class of an IoMT light sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT light sensor are composed of sending light sensed data and its type by a light sensor to other MThings.

4.2.11.2 APIs

Table 11 presents the APIs of an IoMT light sensor.

Table 11 — IoMT light sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MLightSensor()	
Default constructor.	
MLightSensor(string id)	
MLightSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVLight()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `LightSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).

MPEGVSensedDataType	getMPEGVLight(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `LightSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVLight_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVLight()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVLight_Cost(0, "BTC") or getMPEGVLight_Cost(0, "00000001") Ex) getMPEGVLight_Cost(1, "USD") or getMPEGVLight_Cost(1, "10010100")

4.2.12 API for IoMT ambient noise sensor

4.2.12.1 General

This subclause defines a class of an IoMT ambient noise sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT ambient noise sensor are composed of sending ambient noise sensed data and its type by the ambient noise sensor to other MThings.

4.2.12.2 APIs

Table 12 presents the APIs of an IoMT ambient noise sensor.

Table 12 – IoMT ambient noise sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MAmbientNoiseSensor() Default constructor.	
MAmbientNoiseSensor(string id)	

MAmbientNoiseSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVAmbientNoise()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AmbientNoiseSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVAmbientNoise(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AmbientNoiseSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVAmbientNoise_Cost (int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVAmbientNoise()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVAmbientNoise_Cost(0, "BTC") or getMPEGVAmbientNoise_Cost(0, "00000001") Ex) getMPEGVAmbientNoise_Cost(1, "USD") or getMPEGVAmbientNoise_Cost(1, "10010100")

4.2.13 API for IoMT temperature sensor

4.2.13.1 General

This subclause defines a class of an IoMT temperature sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT temperature sensor are composed of sending temperature sensed data and its type, which is acquired by the temperature sensor to other MThings.

4.2.13.2 APIs

Table 13 presents the APIs of an IoMT temperature sensor.

Table 13 — IoMT temperature sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MTemperature()	
Default constructor.	
MTemperature(string id)	
MTemperature(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVTemperature()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `TemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVTemperature(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `TemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVTemperature_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVTemperature()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVTemperature_Cost(0, "BTC") or getMPEGVTemperature_Cost(0, "00000001") Ex) getMPEGVTemperature_Cost(1, "USD") or getMPEGVTemperature_Cost(1, "10010100")

4.2.14 API for IoMT humidity sensor

4.2.14.1 General

This subclause defines a class of an IoMT humidity sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT humidity sensor are composed of sending humidity sensed data and its type, which is acquired by the humidity sensor to other MThings.

4.2.14.2 APIs

Table 14 presents the APIs of an IoMT humidity sensor.

Table 14 — IoMT humidity sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MHumiditySensor()	
Default constructor.	
MHumiditySensor(string id)	
MHumiditySensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVHumidity()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `HumiditySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVHumidity(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `HumiditySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVHumidity_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVHumidity()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVHumidity_Cost(0, "BTC") or getMPEGVHumidity_Cost(0, "00000001") Ex) getMPEGVHumidity_Cost(1, "USD") or getMPEGVHumidity_Cost(1, "10010100")

4.2.15 API for IoMT atmospheric pressure sensor

4.2.15.1 General

This subclause defines a class of an IoMT atmospheric pressure sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT atmospheric pressure sensor are composed of sending atmospheric pressure sensed data and its type, which is acquired by the atmospheric pressure sensor to other MThings.

4.2.15.2 APIs

Table 15 presents the APIs of an IoMT atmospheric pressure sensor.

Table 15 — IoMT atmospheric pressure sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MAtmosphericPressureSensor()	
Default constructor.	
MAtmosphericPressureSensor(string id)	
MAtmosphericPressureSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVAtmosphericPressure()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AtmosphericPressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVAtmosphericPressure(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AtmosphericPressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVAtmosphericPressure_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVAtmosphericPressure()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVAtmosphericPressure_Cost(0, "BTC") or getMPEGVAtmosphericPressure_Cost(0, "00000001") Ex) getMPEGVAtmosphericPressure_Cost(1, "USD") or getMPEGVAtmosphericPressure_Cost(1, "10010100")

4.2.16 API for IoMT velocity sensor

4.2.16.1 General

This subclause defines a class of an IoMT velocity sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT velocity sensor are composed of sending velocity sensed data and its type, which is acquired by the velocity sensor to other MThings.

4.2.16.2 APIs

Table 16 presents the APIs of an IoMT velocity sensor.

Table 16 — IoMT velocity sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MVelocitySensor()	
Default constructor.	
MVelocitySensor(string id)	
MVelocitySensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVVelocity()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `VelocitySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVVelocity(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `VelocitySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVVelocity_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVVelocity()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVVelocity_Cost(0, "BTC") or getMPEGVVelocity_Cost(0, "00000001") Ex) getMPEGVVelocity_Cost(1, "USD") or getMPEGVVelocity_Cost(1, "10010100")

4.2.17 API for IoMT acceleration sensor

4.2.17.1 General

This subclause defines a class of an IoMT acceleration sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT acceleration sensor are composed of sending acceleration sensed data and its type, which is acquired by the acceleration sensor to other MThings.

4.2.17.2 APIs

Table 17 presents the APIs of an IoMT acceleration sensor.

Table 17 — IoMT acceleration sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MAccelerationSensor()	
Default constructor.	
MAccelerationSensor(string id)	
MAccelerationSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVAcceleration()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AccelerationSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVAcceleration(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AccelerationSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVAcceleration_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVAcceleration()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVAcceleration_Cost(0, "BTC") or getMPEGVAcceleration_Cost(0, "00000001") Ex) getMPEGVAcceleration_Cost(1, "USD") or getMPEGVAcceleration_Cost(1, "10010100")

4.2.18 API for IoMT angular acceleration sensor

4.2.18.1 General

This subclause defines a class of an IoMT angular acceleration sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT angular acceleration sensor are composed of sending angular acceleration sensed data and its type, which is acquired by the angular acceleration sensor to other MThings.

4.2.18.2 APIs

Table 18 presents the APIs of an IoMT angular acceleration sensor.

Table 18 — IoMT angular acceleration sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MAngularAccelerationSensor() Default constructor.	
MAngularAccelerationSensor(string id)	
MAngularAccelerationSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVAngularAcceleration()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AngularAccelerationSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVAngularAcceleration(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AngularAccelerationSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVAngularAcceleration_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVAngularAcceleration()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVAngularAcceleration_Cost(0, "BTC") or getMPEGVAngularAcceleration_Cost(0, "00000001") Ex) getMPEGVAngularAcceleration_Cost(1, "USD") or getMPEGVAngularAcceleration_Cost(1, "10010100")

4.2.19 API for IoMT angular velocity sensor

4.2.19.1 General

This subclause defines a class of an IoMT angular velocity sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT angular velocity sensor are composed of sending angular velocity sensed data and its type, which is acquired by the angular velocity sensor to other MThings.

4.2.19.2 APIs

Table 19 presents the APIs of an IoMT angular velocity sensor.

Table 19 — IoMT angular velocity sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MAngularVelocitySensor()	
Default constructor.	
MAngularVelocitySensor(string id)	
MAngularVelocitySensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVAngularVelocity()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AngularVelocitySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVAngularVelocity(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AngularVelocitySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVAngularVelocity_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVAngularVelocity()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVAngularVelocity_Cost(0, "BTC") or getMPEGVAngularVelocity_Cost(0, "00000001") Ex) getMPEGVAngularVelocity_Cost(1, "USD") or getMPEGVAngularVelocity_Cost(1, "10010100")

4.2.20 API for IoMT force sensor

4.2.20.1 General

This subclause defines a class of an IoMT force sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT force sensor are composed of sending force sensed data and its type, which is acquired by the force sensor to other MThings.

4.2.20.2 APIs

Table 20 presents the APIs of an IoMT force sensor.

Table 20 — IoMT force sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MForceSensor() Default constructor.	
MForceSensor(string id)	
MForceSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVForce()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ForceSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVForce(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ForceSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVForce_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVForce()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVForce_Cost(0, "BTC") or getMPEGVForce_Cost(0, "00000001") Ex) getMPEGVForce_Cost(1, "USD") or getMPEGVForce_Cost(1, "10010100")

4.2.21 API for IoMT torque sensor

4.2.21.1 General

This subclause defines a class of an IoMT torque sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT torque sensor are composed of sending torque sensed data and its type, which is acquired by the torque sensor to other MThings.

4.2.21.2 APIs

Table 21 presents the APIs of an IoMT torque sensor.

Table 21 — IoMT torque sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MTorqueSensor()	
Default constructor.	
MTorqueSensor(string id)	
MTorqueSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVTorque()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `TorqueSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVTorque(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `TorqueSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVTorque_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVTorque()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVTorque_Cost(0, "BTC") or getMPEGVTorque_Cost(0, "00000001") Ex) getMPEGVTorque_Cost(1, "USD") or getMPEGVTorque_Cost(1, "10010100")

4.2.22 API for IoMT pressure sensor

4.2.22.1 General

This subclause defines a class of an IoMT pressure sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT pressure sensor are composed of sending pressure sensed data and its type, which is acquired by the pressure sensor to other MThings.

4.2.22.2 APIs

Table 22 presents the APIs of an IoMT pressure sensor.

Table 22 — IoMT pressure sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MPressureSensor()	
Default constructor.	
MPressureSensor(string id)	
MPressureSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVPressure()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `PressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVPressure(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `PressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVPressure_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVPressure()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVPressure_Cost(0, "BTC") or getMPEGVPressure_Cost(0, "00000001") Ex) getMPEGVPressure_Cost(1, "USD") or getMPEGVPressure_Cost(1, "10010100")

4.2.23 API for IoMT motion sensor

4.2.23.1 General

This subclause defines a class of an IoMT motion sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT motion sensor are composed of sending motion sensed data and its type, which is acquired by the motion sensor to other MThings.

4.2.23.2 APIs

Table 23 presents the APIs of an IoMT motion sensor.

Table 23 — IoMT motion sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MMotionSensor()	
Default constructor.	
MMotionSensor(string id)	
MMotionSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVMotion()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MotionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVMotion(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MotionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVMotion_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVMotion()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVMotion_Cost(0, "BTC") or getMPEGVMotion_Cost(0, "00000001") Ex) getMPEGVMotion_Cost(1, "USD") or getMPEGVMotion_Cost(1, "10010100")

4.2.24 API for IoMT intelligent camera sensor

4.2.24.1 General

This subclause defines a class of an IoMT intelligent camera sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT intelligent camera sensor are composed of sending intelligent camera sensed data and its type, which is acquired by the intelligent camera sensor to other MThings.

4.2.24.2 APIs

Table 24 presents the APIs of an IoMT intelligent camera sensor.

Table 24 — IoMT intelligent camera sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MIntelligentCameraSensor()	
Default constructor.	
MIntelligentCameraSensor(string id)	
MIntelligentCameraSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVIntelligentCamera()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `IntelligentCameraType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVIntelligentCamera(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `IntelligentCameraType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVIntelligentCamera_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVIntelligentCamera()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVIntelligentCamera_Cost(0, "BTC") or getMPEGVIntelligentCamera_Cost(0, "00000001") Ex) getMPEGVIntelligentCamera_Cost(1, "USD") or getMPEGVIntelligentCamera_Cost(1, "10010100")

4.2.25 API for IoMT multi interaction point sensor

4.2.25.1 General

This subclause defines a class of an IoMT multi interaction point sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT multi interaction point sensor are composed of sending interaction point sensed data and its type, which is acquired by the multi interaction point sensor to other MThings.

4.2.25.2 APIs

Table 25 presents the APIs of an IoMT multi interaction point sensor.

Table 25 — IoMT multi interaction point sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MMultiInterActionPointSensor()	
Default constructor.	
MMultiInterActionPointSensor(string id)	
MMultiInterActionPointSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVInteractionPoint()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MultiInteractionPointSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVInteractionPoint(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MultiInteractionPointSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVInteractionPoint_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVInteractionPoint()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVInteractionPoint_Cost(0, "BTC") or getMPEGVInteractionPoint_Cost(0, "00000001") Ex) getMPEGVInteractionPoint_Cost(1, "USD") or getMPEGVInteractionPoint_Cost(1, "10010100")

4.2.26 API for IoMT gaze tracking sensor

4.2.26.1 General

This subclause defines a class of an IoMT gaze tracking sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT gaze tracking sensor are composed of sending gaze sensed data and its type, which is acquired by the gaze tracking sensor to other MThings.

4.2.26.2 APIs

Table 26 presents the APIs of an IoMT gaze tracking sensor.

Table 26 — IoMT gaze tracking sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MGazeTrackingSensor()	
Default constructor.	
MGazeTrackingSensor(string id)	
MGazeTrackingSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVGaze()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GazeTrackingSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVGaze(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GazeTrackingSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVGaze_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVGaze()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVGaze_Cost(0, "BTC") or getMPEGVGaze_Cost(0, "00000001") Ex) getMPEGVGaze_Cost(1, "USD") or getMPEGVGaze_Cost(1, "10010100")

4.2.27 API for IoMT wind sensor

4.2.27.1 General

This subclause defines a class of an IoMT wind sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT wind sensor are composed of sending wind sensed data and its type, which is acquired by the wind sensor to other MThings.

4.2.27.2 APIs

Table 27 presents the APIs of an IoMT wind sensor.

Table 27 — IoMT wind sensor API

Nested Classes	
Modifier and Type	**Method and Description**

Constructor	
Constructor and Description	
MWindSensor()	
Default constructor.	
MWindSensor(string id)	
MWindSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVWind()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `WindSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVWind(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `WindSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
Float	getMPEGVWind_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVWind()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVWind_Cost(0, "BTC") or getMPEGVWind_Cost(0, "00000001") Ex) getMPEGVWind_Cost(1, "USD") or getMPEGVWind_Cost(1, "10010100")

4.2.28 API for IoMT altitude sensor

4.2.28.1 General

This subclause defines a class of an IoMT altitude sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT altitude sensor are composed of sending altitude sensed data and its type, which is acquired by the altitude sensor to other MThings.

4.2.28.2 APIs

Table 28 presents the APIs of an IoMT altitude sensor.

Table 28 — IoMT altitude sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MAltitudeSensor()	
Default constructor.	
MAltitudeSensor(string id)	
MAltitudeSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVAltitude()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AltitudeSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVAltitude(string tid)

	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `AltitudeSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVAltitude_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVAltitude()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVAltitude_Cost(0, "BTC") or getMPEGVAltitude_Cost(0, "00000001") Ex) getMPEGVAltitude_Cost(1, "USD") or getMPEGVAltitude_Cost(1, "10010100")

4.2.29 API for IoMT gas sensor

4.2.29.1 General

This subclause defines a class of an IoMT gas sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT gas sensor are composed of gas type sensed data and its type, which is acquired by the gas sensor to other MThings.

4.2.29.2 APIs

Table 29 presents the APIs of an IoMT gas sensor.

Table 29 — IoMT gas sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MGasSensor()	
Default constructor.	
MGasSensor(string id)	

MGasSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVGasType()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GasSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVGasType(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GasSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVGasType_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVGasType()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVGasType_Cost(0, "BTC") or getMPEGVGasType_Cost(0, "00000001") Ex) getMPEGVGasType_Cost(1, "USD") or getMPEGVGasType_Cost(1, "10010100")

4.2.30 API for IoMT dust sensor

4.2.30.1 General

This subclause defines a class of an IoMT dust sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT dust sensor are composed of sending dust value sensed data and its type, which is acquired by the dust sensor to other MThings.

4.2.30.2 APIs

Table 30 presents the APIs of an IoMT dust sensor.

Table 30 — IoMT dust sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MDustSensor()	
Default constructor.	
MDustSensor(string id)	
MDustSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVDustValue()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DustSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVDustValue(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DustSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVDustValue_Cost(int tokenType, string tokenName)

	This function returns the amount of payment (e.g., tokens) to use `getMPEGVDustValue()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVDust_Cost(0, "BTC") or getMPEGVDust_Cost(0, "00000001") Ex) getMPEGVDust_Cost(1, "USD") or getMPEGVDust_Cost(1, "10010100")

4.2.31 API for IoMT bend sensor

4.2.31.1 General

This subclause defines a class of an IoMT bend sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT bend sensor are composed of sending bend value array sensed data and its type, which is acquired by the bend sensor to other MThings.

4.2.31.2 APIs

Table 31 presents the APIs of an IoMT bend sensor.

Table 31 — IoMT bend sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBendSensor()	
Default constructor.	
MBendSensor(string id)	
MBendSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**

Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBendValueArray()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BendSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBendValueArray(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BendSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVBendValueArray_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBendValueArray()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBendValueArray_Cost(0, "BTC") or getMPEGVBendValueArray_Cost(0, "00000001") Ex) getMPEGVBendValueArray_Cost(1, "USD") or getMPEGVBendValueArray_Cost(1, "10010100")

4.2.32 API for IoMT body height sensor

4.2.32.1 General

This subclause defines a class of an IoMT body height sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT body height sensor are composed of sending body height sensed data and its type, which is acquired by the body height sensor to other MThings.

4.2.32.2 APIs

Table 32 presents the APIs of an IoMT body height sensor.

Table 32 — IoMT body height sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	

Constructor and Description	
MBodyHeightSensor()	
Default constructor.	
MBodyHeightSensor(string id)	
MBodyHeightSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBodyHeight()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyHeightSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBodyHeight(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyHeightSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVBodyHeight_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBodyHeight()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBodyHeight_Cost(0, "BTC") or getMPEGVBodyHeight_Cost(0, "00000001") Ex) getMPEGVBodyHeight_Cost(1, "USD") or getMPEGVBodyHeight_Cost(1, "10010100")

4.2.33 API for IoMT body weight sensor

4.2.33.1 General

This subclause defines a class of an IoMT body weight sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT body weight sensor are composed of sending bodyweight sensed data and its type, which is acquired by the bodyweight sensor to other MThings.

4.2.33.2 APIs

Table 33 presents the APIs of an IoMT body weight sensor.

Table 33 — IoMT body weight sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBodyWeightSensor()	
Default constructor.	
MBodyWeightSensor(string id)	
MBodyWeightSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBodyWeight()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyWeightSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBodyWeight(string tid)

	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyWeightSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVBodyWeight_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBodyWeight()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBodyWeight_Cost(0, "BTC") or getMPEGVBodyWeight_Cost(0, "00000001") Ex) getMPEGVBodyWeight_Cost(1, "USD") or getMPEGVBodyWeight_Cost(1, "10010100")

4.2.34 API for IoMT body temperature sensor

4.2.34.1 General

This subclause defines a class of an IoMT body temperature sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT body temperature sensor are composed of sending body temperature sensed data and its type, which is acquired by the body temperature sensor to other MThings.

4.2.34.2 APIs

Table 34 presents the APIs of an IoMT body temperature sensor.

Table 34 — IoMT body temperature sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBodyTemperatureSensor()	
Default constructor.	
MBodyTemperatureSensor(string id)	

MBodyTemperatureSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBodyTemperature()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyTemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBodyTemperature(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyTemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVBodyTemperature_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBodyTemperature()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBodyTemperature_Cost(0, "BTC") or getMPEGVBodyTemperature_Cost(0, "00000001") Ex) getMPEGVBodyTemperature_Cost(1, "USD") or getMPEGVBodyTemperature_Cost(1, "10010100")

4.2.35 API for IoMT body fat sensor

4.2.35.1 General

This subclause defines a class of an IoMT body fat sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT body fat sensor are composed of sending body fat sensed data and its type, which is acquired by the body fat sensor to other MThings.

4.2.35.2 APIs

Table 35 presents the APIs of an IoMT body fat sensor.

Table 35 — IoMT body fat sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBodyFatSensor()	
Default constructor.	
MBodyFatSensor(string id)	
MBodyFatSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBodyFat()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyFatSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBodyFat(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BodyFatSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVBodyFat_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBodyFat()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBodyFat_Cost(0, "BTC") or getMPEGVBodyFat_Cost(0, "00000001") Ex) getMPEGVBodyFat_Cost(1, "USD") or getMPEGVBodyFat_Cost(1, "10010100")

4.2.36 API for IoMT blood type sensor

4.2.36.1 General

This subclause defines a class of an IoMT blood type sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT blood type sensor are composed of sending blood type sensed data and its type, which is acquired by the blood type sensor to other MThings.

4.2.36.2 APIs

Table 36 presents the APIs of an IoMT blood type sensor.

Table 36 — IoMT blood type sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBloodTypeSensor()	
Default constructor.	
MBloodTypeSensor(string id)	
MBloodTypeSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBloodType()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodTypeSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBloodType(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodTypeSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVBloodType_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBloodType()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBloodType_Cost(0, "BTC") or getMPEGVBloodType_Cost(0, "00000001") Ex) getMPEGVBloodType_Cost(1, "USD") or getMPEGVBloodType_Cost(1, "10010100")

4.2.37 API for IoMT blood pressure sensor

4.2.37.1 General

This subclause defines a class of an IoMT blood pressure sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT blood pressure sensor are composed of sending blood pressure sensed data and its type, which is acquired by the blood pressure sensor to other MThings.

4.2.37.2 APIs

Table 37 presents the APIs of an IoMT blood pressure sensor.

Table 37 — IoMT blood pressure sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBloodPressureSensor()	
Default constructor.	
MBloodPressureSensor(string id)	
MBloodPressureSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBloodPressure()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodPressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBloodPressure(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodPressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVBloodPressure_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBloodPressure()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBloodPressure_Cost(0, "BTC") or getMPEGVBloodPressure_Cost(0, "00000001") Ex) getMPEGVBloodPressure_Cost(1, "USD") or getMPEGVBloodPressure_Cost(1, "10010100")

4.2.38 API for IoMT blood sugar sensor

4.2.38.1 General

This subclause defines a class of an IoMT blood sugar sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT blood sugar sensor are composed of sending blood sugar sensed data and its type, which is acquired by the blood sugar sensor to other MThings.

4.2.38.2 APIs

Table 38 presents the APIs of an IoMT blood sugar sensor.

Table 38 — IoMT blood sugar sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBloodSugarSensor()	
Default constructor.	
MBloodSugarSensor(string id)	
MBloodSugarSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBloodSugar()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodSugarSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBloodSugar(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodSugarSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVBloodSugar_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBloodSugar()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBloodSugar_Cost(0, "BTC") or getMPEGVBloodSugar_Cost(0, "00000001") Ex) getMPEGVBloodSugar_Cost(1, "USD") or getMPEGVBloodSugar_Cost(1, "10010100")

4.2.39 API for IoMT blood oxygen sensor

4.2.39.1 General

This subclause defines a class of an IoMT blood oxygen sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT blood oxygen sensor are composed of sending blood oxygen sensed data and its type, which is acquired by the blood oxygen sensor to other MThings.

4.2.39.2 APIs

Table 39 presents the APIs of an IoMT blood sugar sensor.

Table 39 — IoMT blood oxygen sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBloodOxygenSensor()	
Default constructor.	
MBloodOxygenSensor(string id)	
MBloodOxygenSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBloodOxygen()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodOxygenSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBloodOxygen(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BloodOxygenSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVBloodOxygen_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBloodOxygen()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBloodOxygen_Cost(0, "BTC") or getMPEGVBloodOxygen_Cost(0, "00000001") Ex) getMPEGVBloodOxygen_Cost(1, "USD") or getMPEGVBloodOxygen_Cost(1, "10010100")

4.2.40 API for IoMT heart rate sensor

4.2.40.1 General

This subclause defines a class of an IoMT heart rate sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT heart rate sensor are composed of sending heart rate sensed data and its type, which is acquired by the heart rate sensor to other MThings.

4.2.40.2 APIs

Table 40 presents the APIs of an IoMT heart rate sensor.

Table 40 — IoMT heart rate sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MHeartRateSensor() Default constructor.	
MHeartRateSensor(string id)	
MHeartRateSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVHeartRate()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `HeartRateSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVHeartRate(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `HeartRateSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVHeartRate_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVHeartRate()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVHeartRate_Cost(0, "BTC") or getMPEGVHeartRate_Cost(0, "00000001") Ex) getMPEGVHeartRate_Cost(1, "USD") or getMPEGVHeartRate_Cost(1, "10010100")

4.2.41 API for IoMT electrograph sensor

4.2.41.1 General

This subclause defines a class of an IoMT electrograph sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT electrograph sensor are sending electrograph sensed data and its type, which is acquired by the electrograph sensor to other MThings.

4.2.41.2 APIs

Table 41 presents the APIs of an IoMT electrograph sensor.

Table 41 — IoMT electrograph sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MElectrographSensor()	
Default constructor.	
MElectrographSensor(string id)	
MElectrographSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVElectrograph()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ElectrographSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVElectrograph(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ElectrographSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVElectrograph_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVElectrograph()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVElectrograph_Cost(0, "BTC") or getMPEGVElectrograph_Cost(0, "00000001") Ex) getMPEGVElectrograph_Cost(1, "USD") or getMPEGVElectrograph_Cost(1, "10010100")

4.2.42 API for IoMT EEG sensor

4.2.42.1 General

This subclause defines a class of an IoMT EEG sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT EEG sensor are composed of sending EEG sensed data and its type, which is acquired by the EEG sensor to other MThings.

4.2.42.2 APIs

Table 42 presents the APIs of an IoMT EEG sensor.

Table 42 — IoMT EEG sensor API

Nested Classes
Modifier and Type / **Method and Description**
Constructor
Constructor and Description
MEEGSensor() Default constructor.
MEEGSensor(string id)
MEEGSensor(string id, string serverIPAddress, int serverPort)

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVEEG()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EEGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVEEG(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EEGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVEEG_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVEEG()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVEEG_Cost(0, "BTC") or getMPEGVEEG_Cost(0, "00000001") Ex) getMPEGVEEG_Cost(1, "USD") or getMPEGVEEG_Cost(1, "10010100")

4.2.43 API for IoMT ECG sensor

4.2.43.1 General

This subclause defines a class of an IoMT ECG sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT ECG sensor are composed of sending ECG sensed data and its type, which is acquired by the ECG sensor to other MThings.

4.2.43.2 APIs

Table 43 presents the APIs of an IoMT ECG sensor.

Table 43 — IoMT ECG sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MECGSensor()	
Default constructor.	
MECGSensor(string id)	
MECGSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVECG()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ECGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVECG(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ECGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVECG_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVECG()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVECG_Cost(0, "BTC") or getMPEGVECG_Cost(0, "00000001") Ex) getMPEGVECG_Cost(1, "USD") or getMPEGVECG_Cost(1, "10010100")

4.2.44 API for IoMT EMG sensor

4.2.44.1 General

This subclause defines a class of an IoMT EMG sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT EMG sensor are composed of sending EMG sensed data and its type, which is acquired by the EMG sensor to other MThings.

4.2.44.2 APIs

Table 44 presents the APIs of an IoMT EMG sensor.

Table 44 — IoMT EMG sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MEMGSensor()	
Default constructor.	
MEMGSensor(string id)	
MEMGSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVEMG()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EMGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVEMG(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EMGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVEMG_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVEMG()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVEMG_Cost(0, "BTC") or getMPEGVEMG_Cost(0, "00000001") Ex) getMPEGVEMG_Cost(1, "USD") or getMPEGVEMG_Cost(1, "10010100")

4.2.45 API for IoMT EOG sensor

4.2.45.1 General

This subclause defines a class of an IoMT EOG sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT EOG sensor are composed of sending EOG sensed data and its type, which is acquired by the EOG sensor to other MThings.

4.2.45.2 APIs

Table 45 presents the APIs of an IoMT EOG sensor.

Table 45 — IoMT EOG sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MEOGSensor()	
Default constructor.	
MEOGSensor(string id)	
MEOGSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVEOG()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EOGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVEOG(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EOGSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVEOG_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVEOG()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVEOG_Cost(0, "BTC") or getMPEGVEOG_Cost(0, "00000001") Ex) getMPEGVEOG_Cost(1, "USD") or getMPEGVEOG_Cost(1, "10010100")

4.2.46 API for IoMT GSR sensor

4.2.46.1 General

This subclause defines a class of an IoMT GSR sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT GSR sensor are composed of sending GSR sensed data and its type, which is acquired by the GSR sensor to other MThings.

4.2.46.2 APIs

Table 46 presents the APIs of an IoMT GSR sensor.

Table 46 — IoMT GSR sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MGSRSensor()	
Default constructor.	
MGSRSensor(string id)	
MGSRSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVGSRArray()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GSRSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVGSRArray(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GSRSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVGSRArray_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVGSRArray()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVGSRArray_Cost(0, "BTC") or getMPEGVGSRArray_Cost(0, "00000001") Ex) getMPEGVGSRArray_Cost(1, "USD") or getMPEGVGSRArray_Cost(1, "10010100")

4.2.47 API for IoMT bio sensor

4.2.47.1 General

This subclause defines a class of an IoMT biosensor that shall inherit the features of `MSensor` class. The main functions of the IoMT biosensor are composed of sending bio sensed data, which is acquired by the biosensor to other MThings.

4.2.47.2 APIs

Table 47 presents the APIs of an IoMT biosensor.

Table 47 — IoMT biosensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MBioSensor()	
Default constructor.	
MBioSensor(string id)	
MBioSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVBioData()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BioSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVBioData(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BioSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVBioData_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVBioData()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVBioData_Cost(0, "BTC") or getMPEGVBioData_Cost(0, "00000001") Ex) getMPEGVBioData_Cost(1, "USD") or getMPEGVBioData_Cost(1, "10010100")

4.2.48 API for IoMT weather sensor

4.2.48.1 General

This subclause defines a class of an IoMT weather sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT weather sensor are composed of sending weather sensed data and its type, which is acquired by the weather sensor to other MThings.

4.2.48.2 APIs

Table 48 presents the APIs of an IoMT weather sensor.

Table 48 — IoMT weather sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MWeatherSensor()	
Default constructor.	
MWeatherSensor(string id)	
MWeatherSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVWeatherData()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `WeatherSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVWeatherData(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `WeatherSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVWeatherData_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVWeatherData()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVWeatherData_Cost(0, "BTC") or getMPEGVWeatherData_Cost(0, "00000001") Ex) getMPEGVWeatherData_Cost(1, "USD") or getMPEGVWeatherData_Cost(1, "10010100")

4.2.49 API for IoMT facial expression sensor

4.2.49.1 General

This subclause defines a class of an IoMT facial expression sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT facial expression sensor are composed of sending facial expression sensed data and its type, which is acquired by the facial expression sensor to other MThings.

4.2.49.2 APIs

Table 49 presents the APIs of an IoMT facial expression sensor.

Table 49 — IoMT facial expression sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MFacialExpressionSensor()	
Default constructor.	
MFacialExpressionSensor(string id)	
MFacialExpressionSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVFacialExpression()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FacialExpressionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVFacialExpression(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FacialExpressionSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVFacialExpression_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVFacialExpression()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVFacialExpression_Cost(0, "BTC") or getMPEGVFacialExpression_Cost(0, "00000001") Ex) getMPEGVFacialExpression_Cost(1, "USD") or getMPEGVFacialExpression_Cost(1, "10010100")

4.2.50 API for IoMT facial morphology sensor

4.2.50.1 General

This subclause defines a class of an IoMT facial morphology sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT facial morphology sensor are composed of sending facial morphology sensed data and its type, which is acquired by the facial morphology sensor to other MThings.

4.2.50.2 APIs

Table 50 presents the APIs of an IoMT facial morphology sensor.

Table 50 — IoMT facial morphology sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MFacialMorphologySensor()	
Default constructor.	
MFacialMorphologySensor(string id)	
MFacialMorphologySensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVFacialMorphology()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FacialMorphologySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVFacialMorphology(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FacialMorphologySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVFacialMorphology_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use getMPEGVFacialMorphology`()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVFacialMorphology_Cost(0, "BTC") or getMPEGVFacialMorphology_Cost(0, "00000001") Ex) getMPEGVFacialMorphology_Cost(1, "USD") or getMPEGVFacialMorphology_Cost(1, "10010100")

4.2.51 API for IoMT facial expression characteristics sensor

4.2.51.1 General

This subclause defines a class of an IoMT facial expression characteristics sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT facial expression characteristics sensor are composed of sending facial expression characteristics sensed data and its type, which is acquired by the facial expression characteristics sensor to other MThings.

4.2.51.2 APIs

Table 51 presents the APIs of an IoMT facial expression characteristics sensor.

Table 51 — IoMT facial expression characteristics sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MFacialExpressionCharacteristicsSensor()	
Default constructor.	
MFacialExpressionCharacteristicsSensor(string id)	
MFacialExpressionCharacteristicsSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVFacialExpressionCharacteristics()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FacialExpressionCharacteristicsSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVFacialExpressionCharacteristics(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FacialExpressionCharacteristicsSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVFacialExpressionCharacteristics_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVFacialExpressionCharacteristics()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVFacialExpressionCharacteristics_Cost(0, "BTC") or getMPEGVFacialExpressionCharacteristics_Cost(0, "00000001") Ex) getMPEGVFacialExpressionCharacteristics_Cost(1, "USD") or getMPEGVFacialExpressionCharacteristics_Cost(1, "10010100")

4.2.52 API for IoMT geomagnetic sensor

4.2.52.1 General

This subclause defines a class of an IoMT geomagnetic sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT geomagnetic sensor are composed of sending geomagnetic sensed data and its type, which is acquired by the geomagnetic sensor to other MThings.

4.2.52.2 APIs

Table 52 presents the APIs of an IoMT geomagnetic sensor.

Table 52 – IoMT geomagnetic sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MGeomagneticSensor()	
Default constructor.	
MGeomagneticSensor(string id)	
MGeomagneticSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVAzimuth()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GeomagneticSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVAzimuth(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `GeomagneticSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVAzimuth_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVAzimuth()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVAzimuth_Cost(0, "BTC") or getMPEGVAzimuth_Cost(0, "00000001") Ex) getMPEGVAzimuth_Cost(1, "USD") or getMPEGVAzimuth_Cost(1, "10010100")

4.2.53 API for IoMT proximity sensor

4.2.53.1 General

This subclause defines a class of an IoMT proximity sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT proximity sensor are composed of sending proximity sensed data and its type, which is acquired by the proximity sensor to other MThings.

4.2.53.2 APIs

Table 53 presents the APIs of an IoMT proximity sensor.

Table 53 — IoMT proximity sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MProximitySensor()	
Default constructor.	
MProximitySensor(string id)	
MProximitySensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVProximity()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ProximitySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVProximity(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ProximitySensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVProximity_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVProximity()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVProximity_Cost(0, "BTC") or getMPEGVProximity_Cost(0, "00000001") Ex) getMPEGVProximity_Cost(1, "USD") or getMPEGVProximity_Cost(1, "10010100")

4.2.54 API for IoMT switch sensor

4.2.54.1 General

This subclause defines a class of an IoMT switch sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT switch sensor are composed of sending switch sensed data and its type, which is acquired by the switch sensor to other MThings.

4.2.54.2 APIs

Table 54 presents the APIs of an IoMT switch sensor.

Table 54 — IoMT switch sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSwitchSensor()	
Default constructor.	
MSwitchSensor(string id)	
MSwitchSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVSwitch()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SwitchSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVSwitch(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SwitchSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVSwitch_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVSwitch()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVSwitch_Cost(0, "BTC") or getMPEGVSwitch_Cost(0, "00000001") Ex) getMPEGVSwitch_Cost(1, "USD") or getMPEGVSwitch_Cost(1, "10010100")

4.2.55 API for IoMT spectrum camera sensor

4.2.55.1 General

This subclause defines a class of an IoMT spectrum camera sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT spectrum camera sensor are composed of sending spectra sensed data and its type, which is acquired by the spectrum camera sensor to other MThings.

4.2.55.2 APIs

Table 55 presents the APIs of an IoMT spectrum camera sensor.

Table 55 — IoMT spectrum camera sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSpectrumCameraSensor()	
Default constructor.	
MSpectrumCameraSensor(string id)	
MSpectrumCameraSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVSpectra()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SpectrumCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVSpectra(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SpectrumCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVSpectra_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVSpectra()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVSpectra_Cost(0, "BTC") or getMPEGVSpectra_Cost(0, "00000001") Ex) getMPEGVSpectra_Cost(1, "USD") or getMPEGVSpectra_Cost(1, "10010100")

4.2.56 API for IoMT colour camera sensor

4.2.56.1 General

This subclause defines a class of an IoMT colour camera sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT colour camera sensor are composed of sending raw video sensed data and its type, which is acquired by the colour camera sensor to other MThings.

4.2.56.2 APIs

Table 56 presents the APIs of an IoMT colour camera sensor.

Table 56 — IoMT colour camera sensor API

Nested Classes
Modifier and Type	**Method and Description**
Constructor
Constructor and Description
MColourCameraSensor()
Default constructor.
MColourCameraSensor(string id)
MColourCameraSensor(string id, string serverIPAddress, int serverPort)

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVRawVideo()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ColourCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVRawVideo(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ColourCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVRawVideo_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVRawVideo()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVRawVideo_Cost(0, "BTC") or getMPEGVRawVideo_Cost(0, "00000001") Ex) getMPEGVRawVideo_Cost(1, "USD") or getMPEGVRawVideo_Cost(1, "10010100")

4.2.57 API for IoMT depth camera sensor

4.2.57.1 General

This subclause defines a class of an IoMT depth camera sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT depth camera sensor are composed of sending depth camera sensed data and its type, which is acquired by the depth camera sensor to other MThings.

4.2.57.2 APIs

Table 57 presents the APIs of an IoMT depth camera sensor.

Table 57 — IoMT depth camera sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MDepthCameraSensor()	
Default constructor.	
MDepthCameraSensor(string id)	
MDepthCameraSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVDepthCamera()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DepthCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVDepthCamera(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DepthCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVDepthCamera_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVDepthCamera()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVDepthCamera_Cost(0, "BTC") or getMPEGVDepthCamera_Cost(0, "00000001") Ex) getMPEGVDepthCamera_Cost(1, "USD") or getMPEGVDepthCamera_Cost(1, "10010100")

4.2.58 API for IoMT stereo camera sensor

4.2.58.1 General

This subclause defines a class of an IoMT stereo camera sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT stereo camera sensor are composed of sending stereo camera sensed data and its type, which is acquired by the stereo camera sensor to other MThings.

4.2.58.2 APIs

Table 58 presents the APIs of an IoMT stereo camera sensor.

Table 58 — IoMT stereo camera sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MStereoCameraSensor()	
Default constructor.	
MStereoCameraSensor(string id)	
MStereoCameraSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVStereoCamera()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `StereoCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVStereoCamera(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `StereoCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVStereoCamera_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVStereoCamera()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVStereoCamera_Cost(0, "BTC") or getMPEGVStereoCamera_Cost(0, "00000001") Ex) getMPEGVStereoCamera_Cost(1, "USD") or getMPEGVStereoCamera_Cost(1, "10010100")

4.2.59 API for IoMT thermographic camera sensor

4.2.59.1 General

This subclause defines a class of an IoMT thermographic camera sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT thermographic camera sensor are composed of sending thermographic camera sensed data and its type, which is acquired by the thermographic camera sensor to other MThings.

4.2.59.2 APIs

Table 59 presents the APIs of an IoMT thermographic camera sensor.

Table 59 — IoMT thermographic camera sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MThermographicCameraSensor()	
Default constructor.	
MThermographicCameraSensor(string id)	
MThermographicCameraSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVThermographicCamera()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ThermographicCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVThermographicCamera(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ThermographicCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVThermographicCamera_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVThermographicCamera()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVThermographicCamera_Cost(0, "BTC") or getMPEGVThermographicCamera_Cost(0, "00000001") Ex) getMPEGVThermographicCamera_Cost(1, "USD") or getMPEGVThermographicCamera_Cost(1, "10010100")

4.2.60 API for IoMT engine oil temperature sensor

4.2.60.1 APIs

Table 60 presents the APIs of an IoMT engine oil temperature sensor.

Table 60 — IoMT engine oil temperature sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MEngineOilTemperatureSensor()	
Default constructor.	
MEngineOilTemperatureSensor(string id)	
MEngineOilTemperatureSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**

Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVEngineOilTemperature()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EngineOilTemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVEngineOilTemperature(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EngineOilTemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVEngineOilTemperature_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVEngineOilTemperature()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVEngineOilTemperature_Cost(0, "BTC") or getMPEGVEngineOilTemperature_Cost(0, "00000001") Ex) getMPEGVEngineOilTemperature_Cost(1, "USD") or getMPEGVEngineOilTemperature_Cost(1, "10010100")

4.2.61 API for IoMT intake air temperature sensor

4.2.61.1 General

This subclause defines a class of an IoMT intake air temperature sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT intake air temperature sensor are composed of sending intake air temperature sensed data and its type, which is acquired by the intake air temperature sensor to other MThings.

4.2.61.2 APIs

Table 61 presents the APIs of an IoMT intake air temperature sensor.

Table 61 — IoMT intake air temperature sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MIntakeAirTemperatureSensor()	
Default constructor.	
MIntakeAirTemperatureSensor(string id)	
MIntakeAirTemperatureSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVIntakeAirTemperature()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `IntakeAirTemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVIntakeAirTemperature(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `IntakeAirTemperatureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVIntakeAirTemperature_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVIntakeAirTemperature()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVIntakeAirTemperature_Cost(0, "BTC") or getMPEGVIntakeAirTemperature_Cost(0, "00000001") Ex) getMPEGVIntakeAirTemperature_Cost(1, "USD") or getMPEGVIntakeAirTemperature_Cost(1, "10010100")

4.2.62 API for IoMT tire pressure monitor system sensor

4.2.62.1 General

This subclause defines a class of an IoMT tire pressure monitor system sensor that shall inherit the features of `MSensor` class. The IoMT tire pressure monitor system sensor's primary functions are sending the system sensed data and its type, which is acquired by the tire pressure monitor system sensor to other MThings.

4.2.62.2 APIs

Table 62 presents the APIs of an IoMT tire pressure monitor system sensor.

Table 62 — IoMT tire pressure monitor system sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MTirePressureMonitorSystemSensor()	
Default constructor.	
MTirePressureMonitorSystemSensor(string id)	
MTirePressureMonitorSystemSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVTirePressureMonitorSystem()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `TirePressureMonitorSystemSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVTirePressureMonitorSystem(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `TirePressureMonitorSystemSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVTirePressureMonitorSystem_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVTirePressureMonitorSystem()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVTirePressureMonitorSystem_Cost(0, "BTC") or getMPEGVTirePressureMonitorSystem_Cost(0, "00000001") Ex) getMPEGVTirePressureMonitorSystem_Cost(1, "USD") or getMPEGVTirePressureMonitorSystem_Cost(1, "10010100")

4.2.63 API for IoMT distance travelled sensor

4.2.63.1 General

This subclause defines a class of an IoMT distance travelled sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT distance travelled sensor are composed of sending distance travelled sensed data and its type, which is acquired by the distance travelled sensor to other MThings.

4.2.63.2 APIs

Table 63 presents the APIs of an IoMT distance travelled sensor.

Table 63 — IoMT distance travelled sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MDistanceTraveledlSensor()	
Default constructor.	
MDistanceTravelledSensor(string id)	
MDistanceTravelledSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVDistanceTravelled()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DistanceTraveledSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVDistanceTravelled(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `DistanceTraveledSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVDistanceTravelled_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVDistanceTraveled()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVDistanceTraveled_Cost(0, "BTC") or getMPEGVDistanceTraveled_Cost(0, "00000001") Ex) getMPEGVDistanceTraveled_Cost(1, "USD") or getMPEGVDistanceTraveled_Cost(1, "10010100")

4.2.64 API for IoMT speed sensor

4.2.64.1 General

This subclause defines a class of an IoMT speed sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT speed sensor are composed of sending speed sensed data and its type, which is acquired by the speed sensor to other MThings.

4.2.64.2 APIs

Table 64 presents the APIs of an IoMT speed sensor.

Table 64 — IoMT speed sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSpeedSensor()	
Default constructor.	
MSpeedSensor(string id)	
MSpeedSensor(string id, string serverIPAddress, int serverPort)	

Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVSpeed()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SpeedSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVSpeed(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SpeedSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVSpeed_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVSpeed()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVSpeed_Cost(0, "BTC") or getMPEGVSpeed_Cost(0, "00000001") Ex) getMPEGVSpeed_Cost(1, "USD") or getMPEGVSpeed_Cost(1, "10010100")

4.2.65 API for IoMT vehicle speed sensor

4.2.65.1 General

This subclause defines a class of an IoMT vehicle speed sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT vehicle speed sensor are composed of sending vehicle speed sensed data and its type, which is acquired by the vehicle speed sensor to other MThings.

4.2.65.2 APIs

Table 65 presents the APIs of an IoMT vehicle speed sensor.

Table 65 — IoMT vehicle speed sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	

Constructor and Description	
MVehicleSpeedSensor()	
Default constructor.	
MVehicleSpeedSensor(string id)	
MVehicleSpeedSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVVehicleSpeed()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `VehicleSpeedSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVVehicleSpeed(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `VehicleSpeedSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVVehicleSpeed_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVVehicleSpeed()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVVehicleSpeed_Cost(0, "BTC") or getMPEGVVehicleSpeed_Cost(0, "00000001") Ex) getMPEGVVehicleSpeed_Cost(1, "USD") or getMPEGVVehicleSpeed_Cost(1, "10010100")

4.2.66 API for IoMT mass airflow sensor

4.2.66.1 General

This subclause defines a class of an IoMT mass airflow sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT mass airflow sensor are composed of sending mass airflow sensed data and its type, which is acquired by the mass airflow sensor to other MThings.

4.2.66.2 APIs

Table 66 presents the APIs of an IoMT mass airflow sensor.

Table 66 — IoMT mass airflow sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MMassAirFlowSensor()	
Default constructor.	
MMassAirFlowSensor(string id)	
MMassAirFlowSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVMassAirFlow()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MassAirFlowSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).

MPEGVSensedDataType	getMPEGVMassAirFlow(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `MassAirFlowSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVMassAirFlow_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVMassAirFlow()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVMassAirFlow_Cost(0, "BTC") or getMPEGVMassAirFlow_Cost(0, "00000001") Ex) getMPEGVMassAirFlow_Cost(1, "USD") or getMPEGVMassAirFlow_Cost(1, "10010100")

4.2.67 API for IoMT percentage sensor

4.2.67.1 General

This subclause defines a class of an IoMT percentage sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT percentage sensor are composed of sending percentage sensed data and its type, which is acquired by the stereo percentage to other MThings.

4.2.67.2 APIs

Table 67 presents the APIs of an IoMT percentage sensor.

Table 67 — IoMT percentage sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MPercentageSensor()	
Default constructor.	

MPercentageSensor(string id)	
MPercentageSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVPercentage()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `PercentageSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVPercentage(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `PercentageSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVPercentage_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVPercentage()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVPercentage_Cost(0, "BTC") or getMPEGVPercentage_Cost(0, "00000001") Ex) getMPEGVPercentage_Cost(1, "USD") or getMPEGVPercentage_Cost(1, "10010100")

4.2.68 API for IoMT fuel level sensor

4.2.68.1 General

This subclause defines a class of an IoMT fuel level sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT fuel level sensor are composed of sending fuel level sensed data and its type, which is acquired by the fuel level sensor to other MThings.

4.2.68.2 APIs

Table 68 presents the APIs of an IoMT fuel level sensor.

Table 68 — IoMT fuel level sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MFuelLevelSensor()	
Default constructor.	
MFuelLevelSensor(string id)	
MFuelLevelSensor(string id, string serverIPAddress,int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVFuelLevel()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FuelLevelSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVFuelLevel(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FuelLevelSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVFuelLevel_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVFuelLevel()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVFuelLevel_Cost(0, "BTC") or getMPEGVFuelLevel_Cost(0, "00000001") Ex) getMPEGVFuelLevel_Cost(1, "USD") or getMPEGVFuelLevel_Cost(1, "10010100")

4.2.69 API for IoMT manifold absolute pressure sensor

4.2.69.1 General

This subclause defines a class of an IoMT manifold absolute pressure sensor that shall inherit the features of `MSensor` class. The IoMT manifold absolute pressure sensor's primary functions are sending manifold absolute pressure sensed data and its type, which the manifold absolute pressure sensor acquired to other MThings.

4.2.69.2 APIs

Table 69 presents the APIs of an IoMT manifold absolute pressure sensor.

Table 69 — IoMT manifold absolute pressure sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MManifoldAbsolutePressureSensor()	
Default constructor.	
MManifoldAbsolutePressureSensor(string id)	
MManifoldAbsolutePressureSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVManifoldAbsolutePressure()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ManifoldAbsolutePressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVManifoldAbsolutePressure(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ManifoldAbsolutePressureSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVManifoldAbsolutePressure_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVManifoldAbsolutePressure()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVManifoldAbsolutePressure_Cost(0, "BTC") or getMPEGVManifoldAbsolutePressure_Cost(0, "00000001") Ex) getMPEGVManifoldAbsolutePressure_Cost(1, "USD") or getMPEGVManifoldAbsolutePressure_Cost(1, "10010100")

4.2.70 API for IoMT engine RPM sensor

4.2.70.1 General

This subclause defines a class of an IoMT engine RPM sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT engine RPM sensor are composed of sending engine RPM sensed data and its type, which is acquired by the engine RPM sensor to other MThings.

4.2.70.2 APIs

Table 70 presents the APIs of an IoMT engine RPM sensor.

Table 70 — IoMT engine RPM sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MEngineRPMSensor()	
Default constructor.	
MEngineRPMSensor(string id)	
MEngineRPMSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVEngineRPM()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EngineRPMSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVEngineRPM(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EngineRPMSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVEngineRPM_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVEngineRPM()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVEngineRPM_Cost(0, "BTC") or getMPEGVEngineRPM_Cost(0, "00000001") Ex) getMPEGVEngineRPM_Cost(1, "USD") or getMPEGVEngineRPM_Cost(1, "10010100")

4.2.71 API for IoMT Center of Mess sensor

4.2.71.1 General

This subclause defines a class of an IoMT CoM sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT Center of Mess sensor are composed of CoM sensed data and its type, which the Center of Mess sensor acquires to other MThings.

4.2.71.2 APIs

Table 71 presents the APIs of an IoMT CoM sensor.

Table 71 – IoMT CoM sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MCoMSensor()	
Default constructor.	
MCoMSensor(string id)	
MCoMSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVCoM()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `CoMSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVCoM(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `CoMSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVCoM_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVCoM()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVCoM_Cost(0, "BTC") or getMPEGVCoM_Cost(0, "00000001") Ex) getMPEGVCoM_Cost(1, "USD") or getMPEGVCoM_Cost(1, "10010100")

4.2.72 API for IoMT RADAR sensor

4.2.72.1 General

This subclause defines a class of an IoMT RADAR sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT RADAR sensor are composed of sending RADAR sensed data and its type, which is acquired by the RADAR sensor to other MThings.

4.2.72.2 APIs

Table 72 presents the APIs of an IoMT RADAR sensor.

Table 72 — IoMT RADAR sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MRADARSensor()	
Default constructor.	
MRADARSensor(string id)	
MRADARSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVRADAR()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `RADARSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVRADAR(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `RADARSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVRADAR_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVRADAR()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVRADAR_Cost(0, "BTC") or getMPEGVRADAR_Cost(0, "00000001") Ex) getMPEGVRADAR_Cost(1, "USD") or getMPEGVRADAR_Cost(1, "10010100")

4.2.73 API for IoMT array camera sensor

4.2.73.1 General

This subclause defines a class of an IoMT array camera sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT array camera sensor are composed of sending array camera sensed data and its type, which is acquired by the array camera sensor to other MThings.

4.2.73.2 APIs

Table 73 presents the APIs of an IoMT array camera sensor.

Table 73 — IoMT array camera sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MArrayCameraSensor()	
Default constructor.	
MArrayCameraSensor(string id)	
MArrayCameraSensor(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVArrayCamera()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ArrayCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVArrayCamera(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ArrayCameraSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.
float	getMPEGVArrayCamera_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVArrayCamera()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVArrayCamera_Cost(0, "BTC") or getMPEGVArrayCamera_Cost(0, "00000001") Ex) getMPEGVArrayCamera_Cost(1, "USD") or getMPEGVArrayCamera_Cost(1, "10010100")

4.2.74 API for IoMT e-nose sensor

4.2.74.1 General

This subclause defines a class of an IoMT e-nose sensor that shall inherit the features of `MSensor` class. The main functions of the IoMT e-nose sensor are composed of sending enose sensed data and its type, which is acquired by the e-nose sensor to other MThings.

4.2.74.2 APIs

Table 74 presents the APIs of an IoMT e-nose sensor.

Table 74 — IoMT e-nose sensor API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MEnoseSensor()	
Default constructor.	
MEnoseSensor(string id)	
MEnoseSensor(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVSensedDataType	getMPEGVEnose()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EnoseSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5).
MPEGVSensedDataType	getMPEGVEnose(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `EnoseSensorType` from ISO/IEC 23005-5 (MPEG-V Part 5). The `tid` is the transaction ID of payment for using this function.

float	getMPEGVEnose_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getMPEGVEnose()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getMPEGVEnose_Cost(0, "BTC") or getMPEGVEnose_Cost(0, "00000001") Ex) getMPEGVEnose_Cost(1, "USD") or getMPEGVEnose_Cost(1, "10010100")

4.3 APIs for IoMT actuators

4.3.1 General

This subclause defines API classes of IoMT actuators.

4.3.2 MActuator class

4.3.2.1 General

This subclause defines a MActuator class that shall inherit the features of `MThing` class defined in ISO/IEC 23093-2.

4.3.2.2 APIs

Table 75 presents the basic APIs of MActuator.

Table 75 — MActuator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MActuator()	
Default constructor.	
MActuator(string id)	

MActuator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
MPEGVCapabilityType	getMPEGVCapability()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and actuator capabilities from ISO/IEC 23005-2 (MPEG-V Part 2).
int	setMPEGVCommand(MPEGVCommandType mpegvCommand)
	Set a command to an actuator following the specifications of actuator commands from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1, and if the task succeeds, it returns 0, otherwise.
CapabilityListType	getCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a capability list using data formats defined in A.2.
CapabilityListType	getAvailableActuatorCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an available capability list using data formats defined in A.2.2.
CapabilityListType	getAppliedActuatorCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an applied capability list using data formats defined in A.2.2.

4.3.3 API for IoMT speaker

4.3.3.1 General

This subclause defines a class of an IoMT speaker that shall inherit the features of `MActuator` class. The main functions of an IoMT speaker are receiving (a.k.a. setting) an audio URL from other MThings, playing the audio URL and changing the volumes.

4.3.3.2 APIs

Table 76 presents the APIs of an IoMT speaker.

Table 76 — IoMT speaker API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSpeaker()	
Default constructor.	
MSpeaker(string id)	
MSpeaker(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setAudioURL(string url)
	This function receives a URL of an audio source sent by other MThings. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setAudioURL(string tid, string url)
	This function receives a URL of an audio source sent by other MThings. The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setAudioURL_CostPerMinute(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) per minute to use `setAudioURL()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, the function returns −1. Ex) setAudioURL_CostPerMinute(0, "BTC") or setAudioURL_CostPerMinute(0, "00000001") Ex) setAudioURL_CostPerMinute(1, "USD") or setAudioURL_CostPerMinute(1, "10010100")
int	setPlay(ActuationDataType play)
	This function controls the type of play (e.g., stop, play, pause). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setPlay(string tid, ActuationDataType play)
	This function controls the type of play (e.g., stop, play, pause). The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setPlay_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) to use `setPlay()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setPlay_Cost(0, "BTC") or setPlay_Cost(0, "00000001") Ex) setPlay_Cost(1, "USD") or setPlay_Cost(1, "10010100")
int	setVolume(ActuationDataType volume)
	This function adjusts the volume of the sound. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setVolume(string tid, ActuationDataType volume)
	This function adjusts the volume of the sound. The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setVolume_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setVolume()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setVolume_Cost(0, "BTC") or setVolume_Cost(0, "00000001") Ex) setVolume_Cost(1, "USD") or setVolume_Cost(1, "10010100")

4.3.4 API for IoMT display

4.3.4.1 General

This subclause defines a class of an IoMT display that shall inherit the features of `MActuator` class. The main functions of an IoMT display are receiving (a.k.a. setting) a video URL from other MThings, playing the video URL and changing the display characteristics such as resolutions and colours.

4.3.4.2 APIs

Table 77 presents the APIs of an IoMT display.

Table 77 — IoMT display API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MDisplay()	
Default constructor.	
MDisplay(string id)	
MDisplay(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setVideoURL(string videoURL)
	This function receives a URL of a video source sent by other MThings. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setVideoURL(string tid, string videoURL)
	This function receives a URL of a video source sent by other MThings. The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setVideoURL_CostPerMinute(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) per minute to use `setVideoURL()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setVideoURL_CostPerMinute(0, "BTC") or setVideoURL_CostPerMinute(0, "00000001") Ex) setVideoURL_CostPerMinute(1, "USD") or setVideoURL_CostPerMinute(1, "10010100")
int	setPlay(ActuationDataType playType)
	This function controls the type of play (e.g., stop, play, pause). This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setPlay(string tid, ActuationDataType playType)
	This function controls the type of play (e.g., stop, play, pause). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setPlay_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setPlay()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setPlay_Cost(0, "BTC") or setPlay_Cost(0, "00000001") Ex) setPlay_Cost(1, "USD") or setPlay_Cost(1, "10010100")

int	setDisplayResolution(ActuationDataType resolution)
	This function controls the resolution of the video display. The function returns 1; if the task succeeds, it returns 0, otherwise
int	setDisplayResolution(string tid, ActuationDataType resolution)
	This function controls the resolution of the video display. The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setDisplayResolution_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setDisplayResolution()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setDisplayResolution_Cost(0, "BTC") or setDisplayResolution_Cost(0, "00000001") Ex) setDisplayResolution_Cost(1, "USD") or setDisplayResolution_Cost(1, "10010100")
int	setBrightness(ActuationDataType brightness)
	This function controls the brightness of a display. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setBrightness(string tid, ActuationDataType brightness)
	This function controls the brightness of a display. The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setBrightness_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setBrightness()`.If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setBrightness_Cost(0, "BTC") or setBrightness_Cost(0, "00000001") Ex) setBrightness_Cost(1, "USD") or setBrightness_Cost(1, "10010100")
Int	setDisplayMode(int displayType)
	This function sets the display type to either "opaque" or "transparent." If `displayType` is 0, the display type sets to "opaque." If `displayType` is 1, the display mode sets to "transparent." The function returns 1; if the task succeeds, it returns 0, otherwise.

int	setDisplayMode(string tid, int displayType)
	This function sets the display type to either “opaque” or “transparent.” If `displayType` is 0, the display type sets to “opaque.” If `displayType` is 1, the display mode sets to “transparent.” The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setDisplayMode_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setDisplayMode()`. If `tokenType` is 0, it denotes “cryptocurrency,” and if `tokenType` is 1, it denotes “legal tender.” The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setDisplayMode_Cost(0, “BTC”) or setDisplayMode_Cost(0, “00000001”) Ex) setDisplayMode_Cost(1, “USD”) or setDisplayMode_Cost(1, “10010100”)
int	getDisplayMode()
	This function returns the current display type. The function returns 0 if the display is “opaque”; it returns 1 if it is transparent.
int	getDisplayMode(string tid)
	This function returns the current display type. The function returns 0 if the display is “opaque,” returns 1 if the display is transparent. The `tid` is the transaction ID of payment for using this function.
float	getDisplayMode_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getDisplayMode()`.If `tokenType` is 0, it denotes “cryptocurrency,” and if `tokenType` is 1, it denotes “legal tender.” The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getDisplayMode_Cost(0, “BTC”) or getDisplayMode_Cost(0, “00000001”) Ex) getDisplayMode_Cost(1, “USD”) or getDisplayMode_Cost(1, “10010100”)
int	setMPEGVColorCorrector(MPEGVCommandType colorcorrector)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `ColorCorrectionType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.

int	setMPEGVColorCorrector(string tid, MPEGVCommandType colorcorrector)
	This function sets a command to light with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ColorCorrectionType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setMPEGVColorCorrector_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVColorCorrector()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGVColorCorrector_Cost(0, "BTC") or setMPEGVColorCorrector_Cost(0, "00000001") Ex) setMPEGVColorCorrector_Cost(1, "USD") or setMPEGVColorCorrector_Cost(1, "10010100")

4.3.5 API for IoMT camera actuator

4.3.5.1 General

This subclause defines a class of an IoMT camera actuator that shall inherit the features of `MActuator` class. The main functions of an IoMT camera actuator are composed of rotating the camera orientation in 3DoF, zooming, and changing the camera resolutions.

4.3.5.2 APIs

Table 78 presents the APIs of an IoMT camera actuator.

Table 78 — IoMT camera actuator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MCameraActuator()	
Default constructor.	

MCameraActuator(string id)	
MCameraActuator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setCameraOrientation(ActuationDataType orientation)
	This function controls camera orientation. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setCameraOrientation(string tid, ActuationDataType orientation)
	This function receives the orientation sent by other MThings. This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setCameraOrientation_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setCameraOrientation()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setCameraOrientation_Cost(0, "BTC") or setCameraOrientation_Cost(0, "00000001") Ex) setCameraOrientation_Cost(1, "USD") or setCameraOrientation_Cost(1, "10010100")
int	setCameraZoom(ActuationDataType zoom)
	This function controls a camera zoom. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setCameraZoom(string tid, ActuationDataType zoom)
	This function receives the zoom sent by other MThings. This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setCameraZoom_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setCameraZoom()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setCameraZoom_Cost(0, "BTC") or setCameraZoom_Cost(0, "00000001") Ex) setCameraZoom_Cost(1, "USD") or setCameraZoom_Cost(1, "10010100")
int	setCameraResolution(ActuationDataType resolution)
	This function controls the capturing resolution of a video source. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setCameraResolution(string tid, ActuationDataType resolution)
	This function receives the resolution sent by other MThings. This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setCameraResolution_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setCameraResolution()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setCameraResolution_Cost(0, "BTC") or setCameraResolution_Cost(0, "00000001") Ex) setCameraResolution_Cost(1, "USD") or setCameraResolution_Cost(1, "10010100")

4.3.6 API for IoMT hand gesture actuator

4.3.6.1 General

This subclause defines a class of an IoMT hand gesture actuator that shall inherit the features of `MActuator` class. The primary function of an IoMT hand gesture actuator is understanding the received gesture command to activate the actuator accordingly.

4.3.6.2 APIs

Table 79 presents the APIs of an IoMT hand gesture actuator.

Table 79 — IoMT hand gesture actuator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MHandGestureActuator()	
MHandGestureActuator(string id)	
MHandGestureActuator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setHandGestureCommand(ActuationDataType GestureCommand)
	Set a command to an IoMT hand gesture actuator following the specification of the `GestureCommandCS` *following* A.4.3. *The function returns 1; if the task succeeds, it returns 0, otherwise.*
int	setHandGestureCommand(string tid, ActuationDataType GestureCommand)
	Set a command to an IoMT hand gesture actuator following the specification of the `GestureCommandCS` (A.4.3). *The function returns 1; if the task succeeds, it returns 0, otherwise. The* `tid` *is the transaction ID of payment for using this function.*

float	setHandGestureCommand_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setHandGestureCommand()`. *If* `tokenType` *is 0, it denotes "cryptocurrency," and if* `tokenType` *is 1, it denotes "legal tender." The* `tokenName` *is described in a string (e.g., term ID or binary representation) from* `TokenTypeCS` *specified in A.5. If the requested token is not supported, returns –1.* Ex) setHandGestureCommand_Cost(0, "BTC") or setHandGestureCommand_Cost(0, "00000001") Ex) setHandGestureCommand_Cost(1, "USD") or setHandGestureCommand_Cost(1, "10010100")

4.3.7 API for IoMT vibrator

4.3.7.1 General

This subclause defines a class of an IoMT vibrator that shall inherit the features of `MActuator` class. The primary function of an IoMT vibrator is understanding the vibration command described in MPEG-V specifications to activate the IoMT vibrator accordingly.

4.3.7.2 APIs

Table 80 presents the APIs of an IoMT vibrator.

Table 80 — IoMT vibrator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MVibrator()	
Default constructor.	
MVibrator(string id)	
MVibrator(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setVibration(MPEGVCommandType vibration)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `VibrationType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setVibration(string tid, MPEGVCommandType vibration)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `VibrationType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setVibration_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setVibration()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setVibration_Cost(0, "BTC") or setVibration_Cost(0, "00000001") Ex) setVibration_Cost(1, "USD") or setVibration_Cost(1, "10010100")

4.3.8 API for IoMT sprayer

4.3.8.1 General

This subclause defines a class of an IoMT sprayer that shall inherit the features of `MActuator` class. Thus, the primary function of an IoMT sprayer is understanding the sprayer command described in MPEG-V specifications to activate the IoMT sprayer accordingly.

4.3.8.2 APIs

Table 81 presents the APIs of an IoMT sprayer.

Table 81 — IoMT sprayer API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSprayer()	
Default constructor.	
MSprayer(string id)	
MSprayer(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setSprayWater(MPEGVCommandType waterSpray)
	This function sets a command to a water sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SprayerType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setSprayWater(string tid, MPEGVCommandType waterSpray)
	This function sets a command to a water sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `SprayerType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setSprayWater_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setSprayWater()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setSprayWater_Cost(0, "BTC") or setSprayWater_Cost(0, "00000001") Ex) setSprayWater_Cost(1, "USD") or setSprayWater_Cost(1, "10010100")
int	setSprayScent(MPEGVCommandType scentSpray)
	This function sets a command to a scent sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ScentType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setSprayScent(string tid, MPEGVCommandType scentSpray)
	This function sets a command to a scent sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ScentType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setSprayScent_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setSprayScent()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setSprayScent_Cost(0, "BTC") or setSprayScent_Cost(0, "00000001") Ex) setSprayScent_Cost(1, "USD") or setSprayScent_Cost(1, "10010100")
int	setSprayFog(MPEGVCommandType fogSpray)
	This function sets a command to a fog sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FogType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setSprayFog(string tid, MPEGVCommandType fogSpray)
	This function sets a command to a fog sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FogType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setSprayFog_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setSprayFog()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setSprayFog_Cost(0, "BTC") or setSprayFog_Cost(0, "00000001") Ex) setSprayFog_Cost(1, "USD") or setSprayFog_Cost(1, "10010100")
int	setSprayBubble(MPEGVCommandType bubbleSpray)
	This function sets a command to a bubble sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BubbleType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setSprayBubble(string tid, MPEGVCommandType bubbleSpray)
	This function sets a command to a bubble sprayer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `BubbleType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setSprayBubble_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setSprayBubble()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) setSprayBubble_Cost(0, "BTC") or setSprayBubble_Cost(0, "00000001") Ex) setSprayBubble_Cost(1, "USD") or setSprayBubble_Cost(1, "10010100")

4.3.9 API for IoMT light

4.3.9.1 General

This subclause defines a class of an IoMT light that shall inherit the features of `MActuator` class. The primary function of an IoMT light is understanding the light command described in MPEG-V specifications to activate the IoMT light device accordingly.

4.3.9.2 APIs

Table 82 presents the APIs of an IoMT light.

Table 82 — IoMT light API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MLight()	
Default constructor.	
MLight(string id)	
MLight(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setLight(MPEGVCommandType light)
	This function sets a command to a light device with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `LightType` or `FlashType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setLight(string tid, MPEGVCommandType light)
	This function sets a command to a light device with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `LightType` or `FlashType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setLight_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setLight()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setLight_Cost(0, "BTC") or setLight_Cost(0, "00000001") Ex) setLight_Cost(1, "USD") or setLight_Cost(1, "10010100")
int	setArrayedLight(MPEGVCommandType arrayedLight)
	This function sets a command to an arrayed light with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ArrayedLightType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setArrayedLight(string tid, MPEGVCommandType arrayedLight)
	This function sets a command to an arrayed light with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ArrayedLightType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setArrayedLight_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setArrayedLight()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setArrayedLight_Cost(0, "BTC") or setArrayedLight_Cost(0, "00000001") Ex) setArrayedLight_Cost(1, "USD") or setArrayedLight_Cost(1, "10010100")
int	setBrightness(ActuationDataType brightness)
	This function controls the brightness of the light. This function returns 1; if the task succeeds, it returns 0, otherwise.
int	setBrightness(string tid, ActuationDataType brightness)
	This function controls the brightness of the light. This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setBrightness_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setBrightness()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setBrightness_Cost(0, "BTC") or setBrightness_Cost(0, "00000001") Ex) setBrightness_Cost(1, "USD") or setBrightness_Cost(1, "10010100")
int	setMPEGVFlash(MPEGVCommandType flash)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `FlashType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setMPEGVFlash(string tid, MPEGVCommandType flash)
	This function sets a command to a light device with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `FlashType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setMPEGVFlash_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVFlash()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGVFlash_Cost(0, "BTC") or setMPEGVFlash_Cost(0, "00000001") Ex) setMPEGVFlash_Cost(1, "USD") or setMPEGVFlash_Cost(1, "10010100")

4.3.10 API for IoMT heater

4.3.10.1 General

This subclause defines a class of an IoMT heater that shall inherit the features of `MActuator` class. The primary function of an IoMT heater is understanding the heater command described in MPEG-V specifications to activate the IoMT heater accordingly.

4.3.10.2 APIs

Table 83 presents the APIs of an IoMT heater.

Table 83 — IoMT heater API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MHeater()	
Default constructor.	
MHeater(string id)	
MHeater(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setMPEGVHeater(MPEGVCommandType heater)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `HeatingType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setMPEGVHeater(string tid, MPEGVCommandType heater)
	This function sets a command to a heater with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `HeatingType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setMPEGVHeater_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVHeater()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGVHeater_Cost(0, "BTC") or setMPEGVHeater_Cost(0, "00000001") Ex) setMPEGVHeater_Cost(1, "USD") or setMPEGVHeater_Cost(1, "10010100")

4.3.11 API for IoMT cooler

4.3.11.1 General

This subclause defines a class of an IoMT cooler that shall inherit the features of `MActuator` class. The primary function of an IoMT cooler is understanding the cooler command described in MPEG-V specifications to activate the IoMT cooler device accordingly.

4.3.11.2 APIs

Table 84 presents the APIs of an IoMT cooler.

Table 84 — IoMT cooler API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MCooler()	
Default constructor.	
MCooler(string id)	
MCooler(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setMPEGVCooler(MPEGVCommandType cooler)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `CoolingType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setMPEGVCooler(string tid, MPEGVCommandType cooler)
	This function sets a command to a cooler with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `CoolingType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setMPEGVCooler_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVCooler()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGVCooler_Cost(0, "BTC") or setMPEGVCooler_Cost(0, "00000001") Ex) setMPEGVCooler_Cost(1, "USD") or setMPEGVCooler_Cost(1, "10010100")

4.3.12 API for IoMT fan

4.3.12.1 General

This subclause defines a class of an IoMT fan that shall inherit the features of `MActuator` class. Thus, the primary function of an IoMT fan is understanding the fan command described in MPEG-V specifications to activate the IoMT fan device accordingly.

4.3.12.2 APIs

Table 85 presents the APIs of an IoMT fan.

Table 85 — IoMT fan API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MFan()	
Default constructor.	
MFan(string id)	
MFan(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setMPEGVFan(MPEGVCommandType fan)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `WindType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setMPEGVFan(string tid, MPEGVCommandType fan)
	This function sets a command to a fan with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `WindType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setMPEGVFan_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVFan()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGVFan_Cost(0, "BTC") or setMPEGVFan_Cost(0, "00000001") Ex) setMPEGVFan_Cost(1, "USD") or setMPEGVFan_Cost(1, "10010100")

4.3.13 API for IoMT motion chair

4.3.13.1 General

This subclause defines a class of an IoMT motion chair that shall inherit the features of `MActuator` class. The primary function of an IoMT motion chair is understanding the motion chair command described in MPEG-V specifications to activate the IoMT motion chair device accordingly.

4.3.13.2 APIs

Table 86 presents the APIs of an IoMT motion chair.

Table 86 — IoMT motion chair API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MRigidBodyMotion()	
Default constructor.	
MRigidBodyMotion(string id)	
MRigidBodyMotion(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setMPEGVRigidBodyMotion(MPEGVCommandType rigidbodymotion)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `RigidBodyMotionType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setMPEGVRigidBodyMotion(string tid, MPEGVCommandType rigidbodymotion)
	This function sets a command to a motion chair with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `RigidBodyMotionType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setMPEGVRigidBodyMotion_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVRigidBodyMotion()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGVRigidBodyMotion_Cost(0, "BTC") or setMPEGVRigidBodyMotion_Cost(0, "00000001") Ex) setMPEGVRigidBodyMotion_Cost(1, "USD") or setMPEGVRigidBodyMotion_Cost(1, "10010100")

4.3.14 API for IoMT tactile generator

4.3.14.1 General

This subclause defines a class of an IoMT tactile generator that shall inherit the features of `MActuator` class. The primary function of an IoMT tactile generator is understanding the tactile generator command described in MPEG-V specifications to activate the IoMT tactile generator accordingly.

4.3.14.2 APIs

Table 87 presents the APIs of an IoMT tactile generator.

Table 87 — IoMT tactile generator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MTactileGenerator()	
Default constructor.	
MTactileGenerator(string id)	
MTactileGenerator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setMPEGVTactileGenerator(MPEGVCommandType tactile)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `TactileType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setMPEGVTactileGenerator(string tid, MPEGVCommandType tactile)
	This function sets a command to a tactile generator with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `TactileType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.

float	setMPEGVTactileGenerator_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVTactileGenerator()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGVTactileGenerator_Cost(0, "BTC") or setMPEGVTactileGenerator_Cost(0, "00000001") Ex) setMPEGVTactileGenerator_Cost(1, "USD") or setMPEGVTactileGenerator_Cost(1, "10010100")

4.3.15 API for IoMT 3D printer

4.3.15.1 General

This subclause defines a class of an IoMT 3D printer that shall inherit the features of `MActuator` class. Thus, the primary function of an IoMT 3D printer is understanding the 3D printer command described in MPEG-V specifications to activate the IoMT 3D printer accordingly.

4.3.15.2 APIs

Table 88 presents the APIs of an IoMT 3D printer.

Table 88 — IoMT 3D printer API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
M3DPrinter() Default constructor.	
M3DPrinter(string id)	
M3DPrinter(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	setMPEGV3DPrinter(MPEGVCommandType 3Dprinter)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `ThreeDPrinterType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.
int	setMPEGV3DPrinter(string tid, MPEGVCommandType 3Dprinter)
	This function sets a command to a 3D printer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ThreeDPrinterType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setMPEGV3DPrinter_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGVMPEGV3DPrinter()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGV3DPrinter_Cost(0, "BTC") or setMPEGV3DPrinter_Cost(0, "00000001") Ex) setMPEGV3DPrinter_Cost(1, "USD") or setMPEGV3DPrinter_Cost(1, "10010100")
int	setMPEGV3DPrintingColorReproduction(MPEGVCommandType 3Dprintingcolorreproduction)
	The function sets a command with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a device command defined by `ThreeDPrintingColorReproductionType` from ISO/IEC 23005-5 (MPEG-V Part 5). The function returns 1; if the task succeeds, it returns 0, otherwise.

int	setMPEGV3DPrintingColorReproduction(string tid, MPEGVCommandType 3Dprintingcolorreproduction)
	This function sets a command to a 3D printer with a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and sensed data defined by `ThreeDPrintingColorReproductionType` from ISO/IEC 23005-5 (MPEG-V Part 5). This function returns 1; if the task succeeds, it returns 0, otherwise. The `tid` is the transaction ID of payment for using this function.
float	setMPEGV3DPrintingColorReproduction_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `setMPEGV3DPrintingColorReproduction()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) setMPEGV3DPrintingColorReproduction_Cost(0, "BTC") or setMPEGV3DPrintingColorReproduction_Cost(0, "00000001") Ex) setMPEGV3DPrintingColorReproduction_Cost(1, "USD") or setMPEGV3DPrintingColorReproduction_Cost(1, "10010100")

4.4 APIs for IoMT analysers

4.4.1 General

This subclause defines the API classes of IoMT analysers.

4.4.2 MAnalyser class

4.4.2.1 General

This subclause defines a MAnalyser class that shall inherit the features of `MThing` class defined in ISO/IEC 23093-2.

4.4.2.2 APIs

Table 89 presents the basic APIs of MAnalyser.

Table 89 — MAnalyser API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	

Constructor and Description	
MAnalyser()	
Default constructor.	
MAnalyser(string id)	
MAnalyser(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
CapabilityListType	getAnalyserCapabilityList();
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a capability list using data formats defined in subclause A.2.3.
CapabilityListType	getAvailableAnalyserCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an available capability list using data formats defined in subclause A.2.3.
CapabilityListType	getAppliedAnalyserCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an applied capability list using data formats defined in subclause A.2.3.
AnalysedDataType	getAnalysedData()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and analysed data following the specification in subclause A.4.

4.4.3 API for IoMT time synchroniser

4.4.3.1 General

This subclause defines a class of an IoMT time synchroniser that shall inherit the features of `MAnalyser` class. Through APIs provided by IoMT time synchoniser, analysed data can be obtained: two video sources and time offset and token information to use `getSyncedVideo` APIs.

4.4.3.2 APIs

Table 90 presents the APIs of an IoMT time synchroniser.

Table 90 — IoMT time synchroniser API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MTimeSynchroniser()	
Default constructor.	
MTimeSynchroniser(string id)	
MTimeSynchroniser(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getSyncedVideo()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and video sources and audio sources with a time offset following the specification in subclause 8.3.2.

AnalysedDataType	getSyncedVideo(string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and video sources and audio sources with a time offset following the specification in subclause 8.3.2. The `tid` is the transaction ID of payment for using this function.
float	getSyncedVideo_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getSyncedVideo()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getSyncedVideo_Cost(0, "BTC") or getSyncedVideo_Cost(0, "00000001") Ex) getSyncedVideo_Cost(1, "USD") or getSyncedVideo_Cost(1, "10010100")

4.4.4 API for IoMT social event detector

4.4.4.1 General

This subclause defines a class of an IoMT social event detector that shall inherit the features of `MAnalyser` class. APIs provided by IoMT social event detector can be obtained analysed data, which is detected social event and token information to use `getSocialEventName` APIs.

4.4.4.2 APIs

Table 91 presents the APIs of an IoMT social event detector.

Table 91 — IoMT social event detector API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSocialEventDetector()	
Default constructor.	

MSocialEventDetector(string id)	
MSocialEventDetector(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getSocialEventName()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a name of detected social event following the specification in subclause 8.4.2.
AnalysedDataType	getSocialEventName(string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a name of detected social event following the specification in subclause 8.4.2. The `tid` is the transaction ID of payment for using this function.
float	getSocialEventName_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getSocialEventName()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getSocialEventName_Cost(0, "BTC") or getSocialEventName_Cost(0, "00000001") Ex) getSocialEventName _Cost(1, "USD") or getSocialEventName_Cost(1, "10010100")

4.4.5 API for IoMT hand gesture detector

4.4.5.1 General

This subclause defines a class of an IoMT hand gesture detector that shall inherit the features of `MAnalyser` class. Through APIs provided by the IoMT hand gesture detector, other MThings can acquire analysed data from the `MHandGestureDetector` that contain detected hand gestures and token information to use `getDetectedHandContour` APIs.

4.4.5.2 APIs

Table 92 presents the APIs of an IoMT hand gesture detector.

Table 92 — IoMT hand gesture detector API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MHandGestureDetector()	
MHandGestureDetector(string id)	
MHandGestureDetector(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getDetectedHandContour()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and Bezier curves specified in subclause 8.5.
AnalysedDataType	getDetectedHandContour(string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and Bezier curves specified in subclause 8.5. The `tid` is the transaction ID of payment for using this function.

float	getDetectedHandContour_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getDetectedHandContour()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getDetectedHandContour_Cost(0, "BTC") or getDetectedHandContour_Cost(0, "00000001") Ex) getDetectedHandContour_Cost(1, "USD") or getDetectedHandContour_Cost(1, "10010100")

4.4.6 API for IoMT hand gesture recogniser

4.4.6.1 General

This subclause defines a class of an IoMT hand gesture recogniser that shall inherit the features of `MAnalyser` class. Through APIs provided by IoMT hand gesture recogniser, other MThings can acquire analysed data from the `MHandGestureRecognaiser` that contain recognised hand gestures and token information to use `getRecognisedHandPosture` APIs.

4.4.6.2 APIs

Table 93 presents the APIs of an IoMT hand gesture recogniser.

Table 93 — IoMT hand gesture recogniser API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MHandGestureRecogniser()	
MHandGestureRecogniser(string id)	
MHandGestureRecogniser(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getRecognisedHandPosture()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and hand postures specified in subclause 8.6.
AnalysedDataType	getRecognisedHandPosture(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and hand postures specified in subclause 8.6. The `tid` is the transaction ID of payment for using this function.
float	getRecognisedHandPosture_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getRecognisedHandPosture()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getRecognisedHandPosture_Cost(0, "BTC") or getRecognisedHandPosture_Cost(0, "00000001") Ex) getRecognisedHandPosture_Cost(1, "USD") or getRecognisedHandPosture_Cost(1, "10010100")

4.4.7 API for IoMT healthcare information generator

4.4.7.1 General

This subclause defines a class of an IoMT healthcare information generator that shall inherit the features of `MAnalyser` class. Through APIs provided by the IoMT healthcare information generator, other MThings can acquire analysed data from the `MHealthcareInformationGenerator` containing generated healthcare information and token information to use `HealthcareInfoGenerator` APIs.

4.4.7.2 APIs

Table 94 presents the APIs of an IoMT healthcare information generator.

Table 94 — IoMT healthcare information generator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
HealthcareInfoGenerator()	
HealthcareInfoGenerator(string id)	
HealthcareInfoGenerator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	HealthcareInfoGenerator()
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and Healthcare information data type using data formats defined in subclause 8.8.
AnalysedDataType	HealthcareInfoGenerator(string tid)
	The function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and Healthcare information data type using data formats defined in subclause 8.8. The `tid` is the transaction ID of payment for using this function.

float	getHealthcareInfoGenerator_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getHealthcareInfoGenerator_Cost()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getHealthcareInfoGenerator_Cost(0, "BTC") or getHealthcareInfoGenerator_Cost(0, "00000001") Ex) getHealthcareInfoGenerator_Cost(1, "USD") or getHealthcareInfoGenerator_Cost(1, "10010100")

4.4.8 API for IoMT speech recogniser

4.4.8.1 General

This subclause defines a class of an IoMT speech recogniser that shall inherit the features of `MAnalyser` class. The primary function of the IoMT speech recogniser is to obtain analysed data that contain analysed speech and token information to use `getSpeechText` APIs.

4.4.8.2 APIs

Table 95 presents the APIs of an IoMT speech recogniser.

Table 95 — IoMT speech recogniser API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSpeechRecogniser()	
Default constructor.	
MSpeechRecogniser(string id)	
MSpeechRecogniser(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
string	getSpeechText()
	This function returns extracted speech text which is a string, as a result of speech recognition.
string	getSpeechText(string tid)
	This function returns extracted speech text which is a string type, as a result of speech recognition. The `tid` is the transaction ID of payment for using this function.
float	getSpeechText_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getSpeechText()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getSpeechText _Cost(0, "BTC") or getSpeechText_Cost(0, "00000001") Ex) getSpeechText _Cost(1, "USD") or getSpeechText _Cost(1, "10010100")

4.4.9 API for IoMT text to speech converter

4.4.9.1 General

This subclause defines a class of an IoMT text to speech converter that shall inherit the features of `MAnalyser` class. The primary function of the IoMT text to speech converter is to obtain analysed data that contains text and token information to use `getSpeechURL` APIs.

4.4.9.2 APIs

Table 96 presents the APIs of an IoMT text to speech converter.

Table 96 — IoMT text to speech converter API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MTextToSpeech()	
Default constructor.	
MTextToSpeech(string id)	
MTextToSpeech(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods.	
Modifier and Type	**Method and Description**
string	getSpeechURL(string Text)
	This function returns a URL of the speech audio source in string type, which results from text to speech converter. The `Text` is an input string that converts to a speech.
string	getSpeechURL(string tid, string Text)
	This function returns a URL of the speech audio source in string type, which results from text to speech converter. The `Text` is an input string that converts to a speech. The `tid` is the transaction ID of payment for using this function.

float	getSpeechURL_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getSpeechURL()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getSpeechURL _Cost(0, "BTC") or getSpeechURL_Cost(0, "00000001") Ex) getSpeechURL _Cost(1, "USD") or getSpeechURL_Cost(1, "10010100")

4.4.10 API for IoMT question analyser

4.4.10.1 General

This subclause defines a class of an IoMT question analyser that shall inherit the features of `MAnalyser` class. The primary function of the IoMT question analyser is to obtain analysed data which contain analysed questions and token information to use `getUserQuestion` APIs.

4.4.10.2 APIs

Table 97 presents the APIs of an IoMT question analyser.

Table 97 — IoMT question analyser API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
QuestionAnalyser()	
Default constructor.	
QuestionAnalyser(string id)	
QuestionAnalyser(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getUserQuestion()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and user question following the specification in subclause 8.10.
AnalysedDataType	getUserQuestion(string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and user question following the specification in subclause 8.10. The `tid` is the transaction ID of payment for using this function.
float	getUserQuestion_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getUserQuestion()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns –1. Ex) getUserQuestion_Cost(0, "BTC") or getUserQuestion_Cost(0, "00000001") Ex) getUserQuestion_Cost(1, "USD") or getUserQuestion_Cost(1, "10010100")

4.4.11 API for IoMT odour image to scent converter

4.4.11.1 General

This subclause defines a class of an IoMT odour image to scent converter that shall inherit the features of the `MAnalyser` class. The primary function of the IoMT odour image to scent converter is to obtain analysed data, recognised odour images and token information to use `getOdourImageToScentConverterOutput` APIs.

4.4.11.2 APIs

Table 98 presents the APIs of an IoMT odour image to scent converter.

Table 98 — IoMT odour image to scent converter API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MOdourImageToScentConverter()	
Default constructor.	
MOdourImageToScentConverter(string id)	
MOdourImageToScentConverter(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getOdourImageToScentConverterOutput()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and Recognised odour images following the specification in subclause 8.9.
AnalyserDataType	getOdourImageToScentConverterOutput(string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and Recognised odour images following the specification in subclause 8.9. The `tid` is the transaction ID of payment for using this function.

float	getOdourImageToScentConverterOutput_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getOdourImageToScentConverterOutput()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getOdourtImageToScentConverterOutput_Cost(0, "BTC") or getOdourImageToScentConverterOutput _Cost(0, "00000001") Ex) getOdourImageToScentConverterOutput_Cost(1, "USD") or getOdourImageToScentConverterOutput_Cost(1, "10010100")

4.4.12 API for IoMT direction guider

4.4.12.1 General

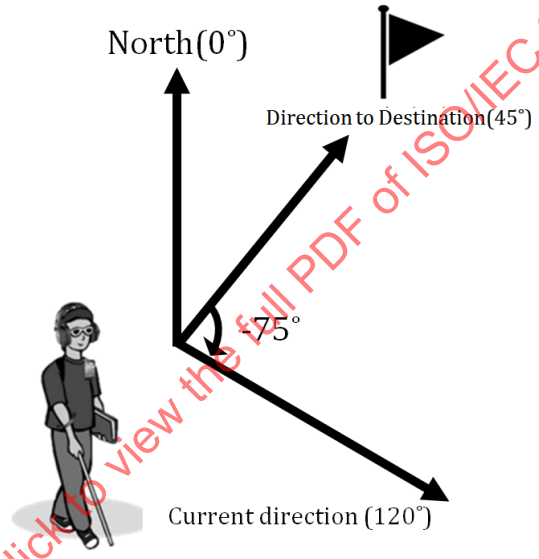
This subclause defines a class of an IoMT direction guider that shall inherit the features of `MAnalyser` class. The primary function of the IoMT director guider is to generate azimuth information and direct guidance for pedestrians to the destination.

4.4.12.2 APIs

Table 99 presents the APIs of an IoMT direction guider.

Table 99 – IoMT direction guider API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MDirectionGuider()	
Default constructor.	
MDirectionGuider(string id)	
MDirectionGuider(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
float	getRelativeAzimuth()
	The function returns a relative clockwise azimuth angle between the current direction and the direction to the destination. (In the case below, the method returns −75.) (Figure 4)  **Figure 4 — Relative azimuth angle**
float	getRelativeAzimuth(string tid)
	The function returns a relative clockwise azimuth angle between the current direction and the direction to the destination. The `tid` is the transaction ID of payment for using this function.
float	getRelativeAzimuth_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getRelativeAzimuth()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getRelativeAzimuth_Cost(0, "BTC") or getRelativeAzimuth_Cost(0, "00000001") Ex) getRelativeAzimuth_Cost(1, "USD") or getRelativeAzimuth_Cost(1, "10010100")

string	getDirectionGuidingText()
	The function returns a human-readable description of the direction to the destination(e.g., “Turn 75 degrees to the left”).
string	getDirectionGuidingText(string tid)
	This function returns a human-readable description of the direction to the destination(e.g., “Turn 75 degrees to the left”). The `tid` is the transaction ID of payment for using this function.
float	getDirectionGuidingText_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getDirectionGuidingText()`. If `tokenType` is 0, it means “cryptocurrency,” and if `tokenType` is 1, it means “legal tender.” The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getDirectionGuidingText_Cost(0, “BTC”) or getDirectionGuidingText_Cost(0, “00000001”) Ex) getDirectionGuidingText_Cost(1, “USD”) or getDirectionGuidingText_Cost(1, “10010100”)

4.4.13 API for IoMT collision coordinator

4.4.13.1 General

This subclause defines a class of an IoMT collision coordinator that shall inherit the features of `MAnalyser` class. The primary function of the IoMT collision coordinator detects obstacles ahead and descriptions about the detected obstacles for pedestrians.

4.4.13.2 APIs

Table 100 presents APIs of an IoMT collision coordinator.

Table 100 — IoMT collision coordinator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MCollisionCoordinator()	

Default constructor.	
MCollisionCoordinator(string id)	
MCollisionCoordinator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
boolean	getDetectObstacle()
	The function returns whether there is an obstacle in front or not.
boolean	getDetectObstacle(string tid)
	The function returns whether there is an obstacle in front or not. The `tid` is the transaction ID of payment for using this function.
float	getDetectObstacle_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getDetectObstacle()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getDetectObstacle_Cost(0, "BTC") or getDetectObstacle_Cost(0, "00000001") Ex) getDetectObstacle_Cost(1, "USD") or getDetectObstacle_Cost(1, "10010100")
string	getTextOfDetectedObstacle()
	The function returns a human-readable statement about obstacle detection.
string	getTextOfDetectedObstacle(string tid)
	The function returns a human-readable statement about obstacle detection. The `tid` is the transaction ID of payment for using this function.

float	getTextOfDetectedObstacle_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getTextOfDetectedObstacle()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getTextOfDetectedObstacle_Cost(0, "BTC") or getTextOfDetectedObstacle_Cost(0, "00000001") Ex) getTextOfDetectedObstacle_Cost(1, "USD") or getTextOfDetectedObstacle_Cost(1, "10010100")

4.4.14 API for IoMT people counter

4.4.14.1 General

This subclause defines a class of an IoMT people counter that shall inherit the features of `MAnalyser` class. This counter discerns the number of people and activity levels (e.g., low, middle, high) in the area.

4.4.14.2 APIs

Table 101 presents the APIs of an IoMT people counter.

Table 101 – IoMT people counter API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MPeopleCounter()	
Default constructor.	
MPeopleCounter(string id)	
MPeopleCounter(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
int	getNumberOfPeople()
	This function returns how many people appear in a video.
int	getNumberOfPeople(string tid)
	This function returns how many people appear in a video. The `tid` is the payment transaction ID of the corresponding service.
int	getNumberOfPeopleInPeriod(long start_time, long end_time)
	This function returns how many people appear in a video in the period. The `start_time` denotes a start time of the period, and the `end_time` denotes an end time of the period. These are expressed in Unix Time in seconds.
int	getNumberOfPeopleInPeriod(string tid, long start_time, long end_time)
	This function returns how many people appear in a video in the period. The `start_time` denotes a start time of the period, and the `end_time` denotes an end time of the period. These are expressed in Unix Time in seconds. The `tid` is the transaction ID of payment for using this function.
int	getActivityLevel()
	This function returns a number between 1 and 3, indicating how crowded the (filmed) place is. (1=low, 2=middle, 3=high).
int	getActivityLevel(string tid)
	This function returns a number between 1 and 3, indicating how crowded the (filmed) place is. (1=low, 2=middle, 3=high). The `tid` is the transaction ID of payment for using this function.

float	getNumberOfPeople_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getNumberOfPeople()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getNumberOfPeople_Cost(0, "BTC") or getNumberOfPeople_Cost(0, "00000001") Ex) getNumberOfPeople_Cost(1, "USD") or getNumberOfPeople_Cost(1, "10010100")
float	getNumberOfPeopleInPeriod_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getNumberOfPeopleInPeriod()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getNumberOfPeopleInPeriod_Cost(0, "BTC") or getNumberOfPeopleInPeriod_Cost(0, "00000001") Ex) getNumberOfPeopleInPeriod_Cost(1, "USD") or getNumberOfPeopleInPeriod_Cost(1, "10010100")
float	getActivityLevel_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getActivityLevel()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getActivityLevel_Cost(0, "BTC") or getActivityLevel_Cost(0, "00000001") Ex) getActivityLevel_Cost(1, "USD") or getActivityLevel_Cost(1, "10010100")

4.4.15 API for IoMT music frequency analyser

4.4.15.1 General

This subclause defines a class of an IoMT music frequency analyser that shall inherit the features of `MAnalyser` class. Through APIs provided by IoMT music frequency analyser, analysed data can be obtained: sampling rate, the number of points at a specific frequency amplitude, and payment information to use `getAnalysedMusicFreq` APIs.

4.4.15.2 APIs

Table 102 presents the APIs of an IoMT music frequency analyser.

Table 102 — IoMT music frequency analyser API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MMusicFrequencyAnalyser()	
Default constructor.	
MMusicFrequencyAnalyser(string id)	
MMusicFrequencyAnalyser(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getAnalysedMusicFrequency()
	This function returns a class (i.e., Java or C++) or a structure(i.e., C) that shall include a returning type (e.g., XML, Binary) and an analysed music frequency according to the audio source and the sampling rate following the specification in subclause 8.11.2.
AnalysedDataType	getAnalysedMusicFrequency (string tid)
	This function returns a class (i.e., Java or C++) or a structure(i.e., C) that shall include a returning type (e.g., XML, Binary) and an analysed music frequency according to the audio source and the sampling rate following the specification in subclause 8.11.2. The `tid` is the transaction ID of payment for using this function.

float	getAnalysedMusicFrequency_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) to use `getAnalysedMusicFrequecy()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getAnalysedMusicFrequency_Cost(0, "BTC") or getAnalysedMusicFrequency_Cost(0, "00000001") Ex) getAnalysedMusicFrequency_Cost(1, "USD") or getAnalysedMusicFrequency_Cost(1, "10010100")

4.4.16 API for IoMT light colour converter

4.4.16.1 General

This subclause defines a class of an IoMT light colour converter that shall inherit the features of `MAnalyser` class. Through APIs provided by IoMT light colour converter, analysed data can be obtained: brightness, saturation, hue value and payment information to use `getAnalysedColourLightInfo` APIs.

4.4.16.2 APIs

Table 103 presents the APIs of an IoMT light colour converter.

Table 103 — IoMT light colour converter API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MLightColorConverter()	
Default constructor.	
MLightColorConverter(string id)	
MLightColorConverter(string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getAnalysedColourLightInfo()
	This function returns a class (i.e., Java or C++) or a structure(i.e., C) that shall include a returning type (e.g., XML, Binary) and an analysed colour light information according to music characteristics following the specification in subclause 8.16.2.
AnalysedDataType	getAnalysedColourLightInfo(string tid)
	This function returns a class (i.e., Java or C++) or a structure(i.e., C) that shall include a returning type (e.g., XML, Binary) and an analysed colour light information according to music characteristics following the specification in subclause 8.16.2. The `tid` is the transaction ID of payment for using this function.
float	getAnalysedColourLightInfo_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) to use `getAnalysedColourLightInfo()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getAnalysedColourLightInfo_Cost(0, "BTC") or getAnalysedColourLightInfo_Cost(0, "00000001") Ex) getAnalysedColourLightInfo_Cost(1, "USD") or getAnalysedColourLightInfo_Cost(1, "10010100")

4.4.17 API for IoMT video content class generator

4.4.17.1 General

This subclause defines a class of an IoMT video content class generator that shall inherit the features of `MAnalyser` class. APIs provided by the IoMT video content class generator can obtain analysed data, which is a content class of video source and payment information to use `getClassofVideoContent` APIs.

4.4.17.2 APIs

Table 104 presents the APIs of an IoMT video content class generator.

Table 104 — IoMT video content class generator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MVideoClassGenerator()	
Default constructor.	
MVideoClassGenerator(string id)	
MVideoClassGenerator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getClassOfVideoContent()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a generated video content class following the specification in subclause 8.12.2.
AnalysedDataType	getClassOfVideoContent(string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a generated video content class following the specification in subclause 8.12.2. The `tid` is the transaction ID of payment for using this function.

float	getClassOfVideoContent_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) to use `getClassofVideoContent()`. If `tokenType` is 0, it means "cryptocurrency," and if `tokenType` is 1, it means "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns −1. Ex) getClassOfVideoContent_Cost(0, "BTC") or getClassOfVideoContent_Cost(0, "00000001") Ex) getClassOfVideoContent_Cost(1, "USD") or getClassOfVideoContent_Cost(1, "10010100")

4.4.18 API for IoMT face region detector

4.4.18.1 General

This subclause defines a class of an IoMT face region detector that shall inherit the features of `MAnalyser` class. APIs provided by the IoMT face region detector can obtain analysed data, which is a list of location and bounding box information related to detected face regions and payment information to use `getFaceRegions` APIs.

4.4.18.2 APIs

Table 105 presents the APIs of an IoMT face region detector.

Table 105 — IoMT face region detector API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MFaceRegionDetector()	
Default constructor.	
MFaceRegionDetector (string id)	
MFaceRegionDetector (string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getFaceRegions()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) and information of detected face(s) following the specification in subclause 8.13.2.
AnalysedDataType	getFaceRegions (string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) information of detected face(s) following the specification in subclause 8.13.2. The `tid` is the transaction ID of payment for using this function.
Float	getFaceRegions_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) to use `getFaceRegions()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getFaceRegions_Cost(0, "BTC") or getFaceRegions_Cost(0, "00000001") Ex) getFaceRegions_Cost(1, "USD") or getFaceRegions_Cost(1, "10010100")

4.4.19 API for IoMT face verifier

4.4.19.1 General

This subclause defines a class of an IoMT face verifier that shall inherit the features of `MAnalyser` class. Through APIs provided by IoMT face verifier, analysed data can be obtained, which is verification result and the confidence of face image and payment information to use `getFaceVerification` APIs.

4.4.19.2 APIs

Table 106 presents the APIs of an IoMT face verifier.

Table 106 — IoMT face verifier API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MFaceVerifier()	
Default constructor.	
MFaceVerifier (string id)	
MFaceVerifier (string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
string refImageURL	URL of face image to be compared
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getFaceVerification (<List> string refImageSetURL)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) and a face verification result following the specification in subclause 8.14.2. Argument `refImageSetURL` is URL information of a set of images to be compared as a reference.
AnalysedDataType	getFaceVerification (<List> string refImageSetURL, string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) and a face verification result following the specification in subclause 8.14.2. The `refImageSetURL` is the URL information of a set of images to be compared as a reference, and the `tid` is the transaction ID of payment for using this function.

float	getFaceVerification_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) to use `getFaceVerification()`. If `tokenType` is 0, it denotes "cryptocurrency," and if `tokenType` is 1, it denotes "legal tender." The `tokenName` is described in a string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getFaceVerification_Cost(0, "BTC") or getFaceVerification_Cost(0, "00000001") Ex) getFaceVerification_Cost(1, "USD") or. getFaceVerification_Cost(1, "10010100")

4.4.20 API for IoMT security alert generator

4.4.20.1 General

This subclause defines a class of an IoMT security alert generator that shall inherit the features of `MAnalyser` class. Through APIs provided by the IoMT security alert generator, the video URL can be obtained for security alerting and payment information to use `getSecurityAlertVideoURL` APIs.

4.4.20.2 APIs

Table 107 presents the APIs of an IoMT security alert generator.

Table 107 IoMT security alert generator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSecurityAlertGenerator ()	
Default constructor.	
MSecurityAlertGenerator (string id)	
MSecurityAlertGenerator (string id, string serverIPAddress, int serverPort)	

Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
string	getSecurityAlertVideoURL ()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) and a security alert video URL.
string	getSecurityAlertVideoURL (string tid)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) and a security alert video URL. The `tid` is the transaction ID of payment for using this function.
float	getSecurityAlertVideoURL _Cost(int tokenType, string tokenName)
	This function returns the amount of payment (e.g., tokens) to use `getSecurityAlertVideoURL`(). If tokenType is 0, it means "Crypto Currency," and if `tokenType` is 1, it means "Legal Tender." The `tokenName` is described in the string (e.g., term ID or binary representation) from `TokenTypeCS` specified in A.5. If the requested token is not supported, returns -1. Ex) getSecurityAlertVideoURL _Cost (0, "BTC") or getSecurityAlertVideoURL _Cost (0, "00000001") Ex) getSecurityAlertVideoURL _Cost (1, "USD") or getSecurityAlertVideoURL _Cost (1, "10010100")

4.4.21 API for IoMT security title generator

4.4.21.1 General

This subclause defines a class of an IoMT security title generator that shall inherit the features of `MAnalyser` class. APIs provided by the IoMT security alert generator can obtain analysed data that are detected time, location(latitude, longitude), and security event title and payment information to use `getGeneratedSecurityTitle` APIs.

4.4.21.2 APIs

Table 108 presents the APIs of an IoMT security title generator.

Table 108 IoMT security title generator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MSecurityTitleGenerator ()	
Default constructor.	
MSecurityTitleGenerator (string id)	
MSecurityTitleGenerator (string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
AnalysedDataType	getGereratedSecurityTitle (string URL, SensedDataType detectedTime, , float latitude, float longitude)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) and a generated security title following the specification in subclause 8.15.2. The `URL`, `detectedTime, latitude, and longitude` are parameters for generating a security title.
AnalysedDataType	getGereratedSecurityTitle (string tid, string URL, SensedDataType detectedTime, float latitude, float longitude)
	This function returns a class (i.e., Java or C++) or a structure (i.e., C), which include a returning type (e.g., XML, Binary) and a generated security title following the specification in subclause 8.15.2. The `tid` is the transaction ID of payment for using this function. The `URL`, `detectedTime, latitude, and longitude` are parameters for generating a security title.

float	getGeneratedSecurityTitle_Cost(int tokenType, string tokenName)
	This function returns the amount of payment (i.e., tokens) to use `getGeneratedSecurityTitle`(). If tokenType is 0, it means "Crypto Currency," and if `tokenType` is 1, it means "Legal Tender." The tokenName is described in a string (e.g., term ID or binary representation) from TokenTypeCS specified in A.5. If the requested token is not supported, returns -1. Ex) getGeneratedSecurityTitle_Cost (0, "BTC") or getGeneratedSecurityTitle_Cost (0, "00000001") Ex) getGeneratedSecurity_Cost (1, "USD") or getGeneratedSecurityTitle_Cost (1, "10010100")

4.5 APIs for IoMT storages

4.5.1 General

This subclause defines API classes of IoMT storage.

4.5.2 MStorage class

4.5.2.1 General

This subclause defines an MStorage class that shall inherit the features of `MThing` class defined in ISO/IEC 23093-2.

4.5.2.2 APIs

Table 109 presents the basic APIs of MStorage.

Table 109 — MStorage API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MStorage()	
Default constructor.	
MStorage(string id)	

MStorage(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
string	saveVideo(string videoURL)
	This function saves a received video source with a URL. The function returns a unique file ID.
string	saveImage(string imageURL)
	This function saves a received image source with a URL. The function returns a unique file ID.
string	saveAudio(string audioURL)
	This function saves a received audio source with a URL. The function returns a unique file ID.
string	saveAnalysedDataURL(string analysedDataURL)
	This function saves a received analysed data source with a URL. The function returns a unique file ID.
string	saveAnalysedData(AnalysedDataType analysedData)
	This function saves a received analysed data source. The function returns a unique file ID.
string	getVideo(string fileID)
	This function returns a video source with its URL, which matches the specified file ID.
string	getImage(string fileID)
	This function returns an image source with its URL, which matches the specified file ID.
string	getAudio(string fileID)
	This function returns an audio source with its URL, which matches the specified file ID.
string	getAnalysedDataURL(string fileID)
	This function returns a URL of analysed data that matches with the specified file ID.

AnalysedDataType	getAnalysedData(string fileID)
	This function returns analysed data following the data formats defined in A.4, which matches the specified file ID.
int	updateVideo(string videoURL, string fileID)
	This function updates an existing video source with the specified file ID to the video source with its URL. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	updateImage(string imageURL, string fileID)
	This function updates an existing image source with the specified file ID to the image source with its URL. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	updateAudio(string audioURL, string fileID)
	This function updates an existing audio source with the specified file ID to the audio source with its URL. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	updateAnalysedDataURL(string analysedDataURL, string fileID)
	This function updates existing analysed data with the specified file ID with its URL. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	updateAnalysedData(AnalysedDataType analysedData, string fileID)
	This function updates analysed data with the specified file ID to the analysed data following the data formats defined in A.4. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	deleteFile(string fileID)
	This function deletes an existing file with the specified file ID. The function returns 1; if the task succeeds, it returns 0, otherwise.

4.6 APIs for IoMT managers

4.6.1 General

This subclause defines API classes of IoMT managers.

4.6.2 MManager class

4.6.2.1 General

This subclause defines an MManager class that shall inherit the features of `MThing` class defined in ISO/IEC 23093-2.

4.6.2.2 APIs

Table 110 presents the basic APIs of MManager.

Table 110 — MManager API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MManager()	
Default constructor.	
MManager(string id)	
MManager(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
List<MThingInfoType>	getMThingListByCapability(string MThingType, string capability)
	This function returns a list of MThings among the registered MThings in the MManager. The `MThingType` and `capability` (i.e., formal parameters) can be described in a string (e.g., term ID or binary representation) from the `MThingTypeCS` and `capabilityCS` defined in Annex A.1 (Table A.1) and Annex A.2.
CapabilityListType	getManagerCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type(e.g., XML, Binary) and a capability list specified in A.2.5 (Table A.6).
CapabilityListType	getAvailableManagerCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an available capability list specified in A.2.5.

CapabilityListType	getAppliedManagerCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an applied capability list specified in A.2.5.
int	registerMThing(MThingInfoType MThingInfo)
	This function registers an MThing to the MManager. The function returns 1; if the task succeeds, it returns 0, otherwise.
int	unregisterMThing(MThingInfoType MThingInfo)
	This function unregisters an MThing from the MManager. The function returns 1; if the task succeeds, it returns 0, otherwise.
List<MThingInfoType>	getRegisteredMThingList()
	This function returns a list of MThings registered in the MManager.

4.7 APIs for IoMT aggregators

4.7.1 General

This subclause defines API classes of IoMT aggregators.

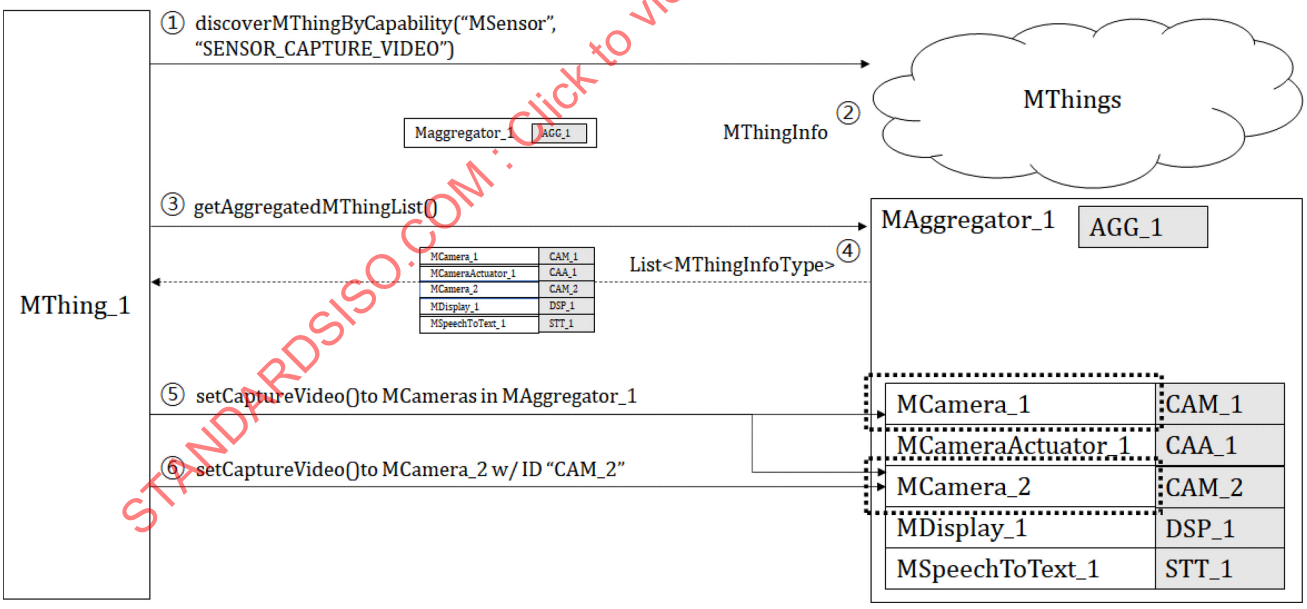


Figure 5 — Process of activating `setCaptureVideo()` to cameras in a MAggregator

Figure 5 shows a process of activating `setCaptureVideo()` to cameras in a MAggregator. A MThing (i.e., MThing_1) can search for MSensors with a specific capability (e.g., SENSOR_CAPTURE_VIDEO) with the function `discoverMThingByCapability()` (Figure 5, item 1). Suppose that the IoT network or server returns MThing information of MAggregator_1 with its ID (Figure 5, item 2) as the search result. It can be possible because the MAggregator_1 possesses MSensors with the video capturing capability.

The MThing_1 can ask for a list of MThings that constitute the MAggregator_1 with the `getAggregatedMThingList()` (Figure 5, item 3). The MAggregator_1 returns a list of aggregated MThing information (Figure 5, item 4), and the MThing_1 can send a command `setCaptureVideo()` to all cameras using the information received (Figure 5, item 5). The MThing_1 can also send a command `setCaptureVideo()` to a specific camera using ID information (Figure 5, item 6).

Figure 6 and Figure 7 show the example codes of how to access all cameras in a MAggregator (Figure 5, item 5) and access a specific camera with ID in a MAggregator (Figure 5, item 6).

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

        MAggregator mag01 = new MAggregator(connectedMAggregator);
        MCamera tempCamera = new MCamera();

        List<MThing> aggList = parseDataToList(mag01.getAggregatedMThingList());

        //command setCaptureVideo to MCameras
        for(int i=0; i<aggList.size(); i++){
            if(aggList.get(i) instanceof MCamera){
                tempCamera = (MCamera) aggList.get(i);
                tempCamera.setCaptureVideo();
            }
        }
}
```

Figure 6 — An example code of accessing all cameras in a MAggregator

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

        MAggregator mag01 = new MAggregator(connectedMAggregator);
        MCamera tempCamera = new MCamera();

        List<MThing> aggList = parseDataToList(mag01.getAggregatedMThingList());

        //command setCaptureVideo to an MCamera with ID "CAM_2"
        for(int i=0; i<aggList.size(); i++){
            if(aggList.get(i).getIdRef().equals("CAM_2")){
                tempCamera = (MCamera) aggList.get(i);
                tempCamera.setCaptureVideo();
                break;
            }
        }
}
```

Figure 7 — An example code of accessing a specific camera in a MAggregator

4.7.2 MAggregator class

4.7.2.1 General

This subclause defines a MAggregator class that shall inherit the features of `MThing` class defined in ISO/IEC 23093-2. The MAggregator APIs return the composite MThings and capabilities of the aggregator.

4.7.2.2 APIs

Table 111 presents the basic APIs of MAggregator.

Table 111 — MAggregator API

Nested Classes	
Modifier and Type	**Method and Description**
Constructor	
Constructor and Description	
MAggregator()	
Default constructor.	
MAggregator(string id)	
MAggregator(string id, string serverIPAddress, int serverPort)	
Fields	
Modifier and Type	**Field and Description**
Methods	
Modifier and Type	**Method and Description**
List<MThingInfoType>	getAggregatedMThingList()
	This function returns a list of MThings that consists of a MAggregator.
void	setAggregatedMThingList(List<MThingInfoType> aggregatedMThingList)
	This function sets a MAggregator with the designated list of MThings.
CapabilityListType	getAggregatorCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and a capability list specified in A.2.6.

CapabilityListType	getAvailableAggregatorCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an available capability list specified in A.2.6.
CapabilityListType	getAppliedAggregatorCapabilityList()
	This function returns a class (i.e., Java or C++) or a structure (i.e., C) that shall include a returning type (e.g., XML, Binary) and an applied capability list specified in A.2.6.

4.8 Return type class

4.8.1 General

This subclause specifies return class types of data exchanged between MThings. For easy industrial adoption, the return type classes are provided in JAVA, C++, and C.

4.8.2 MPEGVCapabilityType

4.8.2.1 General

This subclause contains the capability description class types of MPEG-V sensors and actuators following ISO/IEC 23005-2 (MPEG-V Part 2) capability specification. Figure 8 shows how the one MThing (i.e., "MThing2") requests MPEG-V capability description of another MThing (i.e., "MThing1") by invoking the `getMPEGVCapbility()` (Figure 8, items 1 and 2). The "MThing1" then sends back its MPEG-V capability description using the return class type (i.e., `MPEGVCapabilityType`) to the "MThing2" (Figure 8, item 7). Only steps 1, 2 and 7 are visible between MThings. Other steps are hidden processes performed inside the MThings.

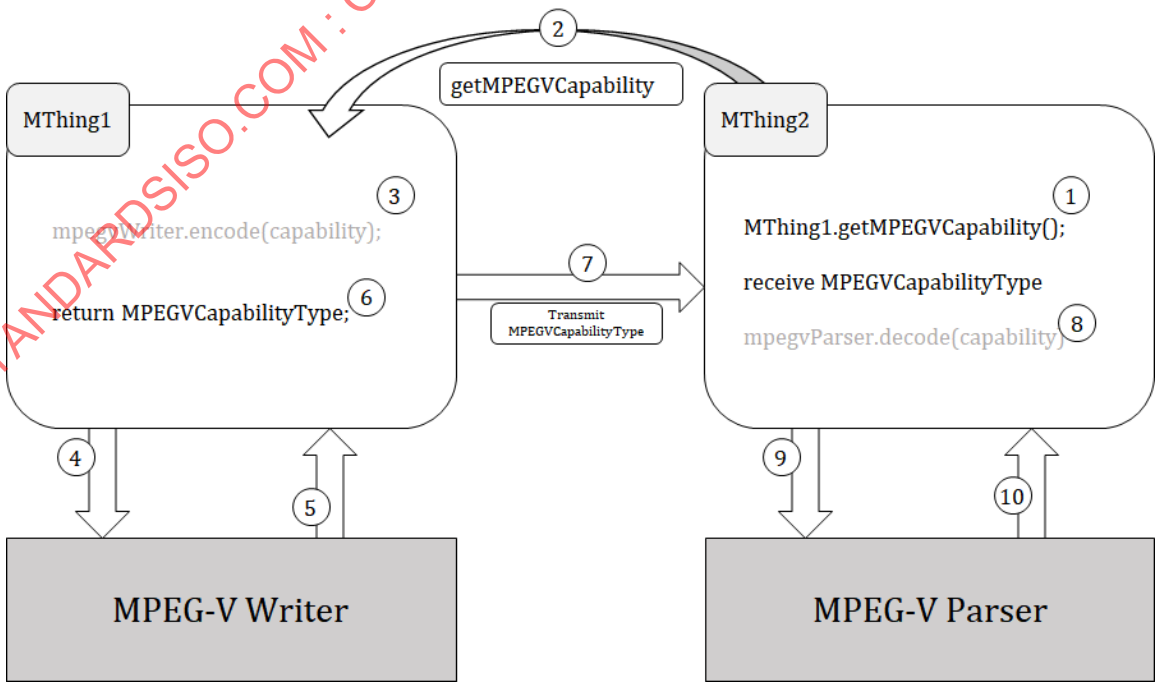


Figure 8 — Requesting a capability description of MThing described following ISO/IEC 23005-2 (MPEG-V Part 2) specification

4.8.2.2 JAVA class

Table 112 presents return class types of MPEG-V capability information in JAVA.

Table 112 — Return class types of MPEG-V capability information in JAVA

Constructor	
Constructor and Description	
MPEGVCapabilityType()	
Default constructor.	
MPEGVCapabilityType(int type, string mpegvData)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of capability metadata. 0 = None, 1 = XML, 2 = Binary
string	mpegvData
	It is capability information that shall follow the specifications of capabilities from ISO/IEC 23005-2 (MPEG-V Part 2) (i.e., sensor capability, device capability).
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)

string	getMPEGVData()
void	setMPEGVData(string mpegvData)

4.8.2.3 C++ class

Table 113 presents return class types of MPEG-V capability information in C++.

Table 113 — Return class types of MPEG-V capability information in C++

Constructor and Destructor	
Constructor and Destructor Description	
MPEGVCapabilityType()	
Default constructor.	
MPEGVCapabilityType(int type, string mpegvData)	
~MPEGVCapabilityType()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of capability data. 0 = None, 1 = XML, 2 = Binary
string	mpegvData
	It is capability information that shall follow the specifications of capabilities from ISO/IEC 23005-2 (MPEG-V Part 2) (i.e., sensor capability, device capability).

Public Member Functions	
Modifier and Type	**Public Member Functions and Description**
int	getType()
void	setType(int type)
string	getMPEGVData()
void	setMPEGVData(string mpegvData)

4.8.2.4 C structure

Table 114 presents return class types of MPEG-V capability information in C.

Table 114 — Return class types of MPEG-V capability information in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of capability data. 0 = None, 1 = XML, 2 = Binary
char*	mpegvData
	It is capability information that shall follow the specifications of capabilities from ISO/IEC 23005-2 (MPEG-V Part 2) (i.e., sensor capability, device capability).

4.8.3 MPEGVSensedDataType

4.8.3.1 General

This subclause contains the sensed data description class types of MPEG-V sensors following ISO/IEC 23005-5 (MPEG-V Part 5) sensed data specification.

4.8.3.2 JAVA class

Table 115 presents return class types of MPEG-V sensed data in JAVA.

Table 115 — Return class types of MPEG-V sensed data in JAVA

Constructor	
Constructor and Description	
MPEGVSensedDataType()	
Default constructor.	
MPEGVSensedDataType(int type, string mpegvData)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
string	mpegvData
	Sensed data of an IoMT sensor shall follow the sensed data specifications from ISO/IEC 23005-5 (MPEG-V Part 5).
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)

string	getMPEGVData()
void	setMPEGVData(string mpegvData)

4.8.3.3 C++ class

Table 116 presents return class types of MPEG-V sensed data in C++.

Table 116 — Return class types of MPEG-V sensed data in C++

Constructor and Destructor	
Constructor and Destructor Description	
MPEGVSensedDataType()	
Default constructor.	
MPEGVSensedDataType(int type, string mpegvData)	
~MPEGVSensedDataType()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
string	mpegvData
	Sensed data of an IoMT sensor shall follow the sensed data specifications from ISO/IEC 23005-5 (MPEG-V Part 5).
Public Member Functions	
Modifier and Type	**Public Member Functions and Description**

int	getType()
void	setType(int type)
string	getMPEGVData()
void	setMPEGVData(string mpegvData)

4.8.3.4 C structure

Table 117 presents return class types of MPEG-V sensed data in C.

Table 117 — Return class types of MPEG-V sensed data in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
char*	mpegvData
	Sensed data of an IoMT sensor shall follow the sensed data specifications from ISO/IEC 23005-5 (MPEG-V Part 5).

4.8.4 MPEGVCommandType

4.8.4.1 General

This subclause contains the command description class types for MPEG-V actuators following ISO/IEC 23005-5 (MPEG-V Part 5) command specification.

4.8.4.2 JAVA class

Table 118 presents return class types of MPEG-V command in JAVA.

Table 118 — Return class types of MPEG-V command in JAVA

Constructor	
Constructor and Description	
MPEGVCommandType()	
Default constructor.	
MPEGVCommandType(int type, string mpegvData)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
string	mpegvData
	It is a command that shall follow the specifications of device commands from ISO/IEC 23005-5 (MPEG-V Part 5).
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)
string	getMPEGVData()
void	setMPEGVData(string mpegvData)

4.8.4.3 C++ class

Table 119 presents return class types of MPEG-V command in C++.

Table 119 — Return class types of MPEG-V command in C++

Constructor and Destructor	
Constructor and Destructor Description	
MPEGVCommandType()	
Default constructor.	
MPEGVCommandType(int type, string mpegvData)	
~MPEGVCommandType ()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
string	mpegvData
	It is a command that shall follow the specifications of device commands from ISO/IEC 23005-5 (MPEG-V Part 5).
Public Member Functions	
Modifier and Type	**Public Member Functions and Description**
int	getType()
void	setType(int type)

string	getMPEGVData()
void	setMPEGVData(string mpegvData)

4.8.4.4 C structure

Table 120 presents return class types of MPEG-V command in C.

Table 120 — Return class types of MPEG-V command in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
char*	mpegvData
	It is a command that shall follow the specifications of device commands from ISO/IEC 23005-5 (MPEG-V Part 5).

4.8.5 IoMT SensedDataType

4.8.5.1 General

This subclause contains the sensed data description class types of IoMT sensors following the data format specification in subclauses 5 and 6.

4.8.5.2 JAVA class

Table 121 presents return class types of IoMT sensed data in JAVA.

Table 121 — Return class types of IoMT sensed data in JAVA

Constructor	
Constructor and Description	
SensedDataType()	
Default constructor.	
SensedDataType(int type, string IoMTData)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	Sensed data of an IoMT sensor shall follow `MTDL` and `MSOV` defined in subclauses 5 and 6.
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)
string	getIoMTData()
void	setIoMTData(string IoMTData)

4.8.5.3 C++ class

Table 122 presents return class types of IoMT sensed data in C++.

Table 122 — Return class types of IoMT sensed data in C++

Constructor and Destructor	
Constructor and Destructor Description	
SensedDataType()	
Default constructor.	
SensedDataType(int type, string IoMTData)	
~SensedDataType()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	Sensed data of an IoMT sensor shall follow `MTDL` and `MSOV` defined in subclauses 5 and 6.
Public Member Functions	
Modifier and Type	**Public Member Functions and Description**
int	getType()
void	setType(int type)

string	getIoMTData()
void	setIoMTData(string IoMTData)

4.8.5.4 C structure

Table 123 presents return class types of IoMT sensed data in C.

Table 123 — Return class types of IoMT sensed data in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of sensed data. 0 = None, 1 = XML, 2 = Binary
Char*	IoMTData
	Sensed data of an IoMT sensor shall follow `MTDL` and `MSOV` defined in subclauses 5 and 6.

4.8.6 IoMT ActuationDataType

4.8.6.1 General

This subclause contains the actuation data class types of IoMT actuators following the data format specification in subclauses 5 and 7.

4.8.6.2 JAVA class

Table 124 presents return class types of IoMT actuation data in JAVA.

Table 124 — Return class types of IoMT actuation data in JAVA

Constructor	
Constructor and Description	
ActuationDataType()	
Default constructor.	
ActuationDataType(int type, string IoMTData)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of actuation data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	Actuation data from another MThing shall follow `MTDL` and `MACV` defined in subclauses 5 and 7.
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)
string	getIoMTData()
void	setIoMTData(string IoMTData)

4.8.6.3 C++ class

Table 125 presents return class types of IoMT actuation data in C++.

Table 125 — Return class types of IoMT actuation data in C++

Constructor and Destructor	
Constructor and Destructor Description	
ActuationDataType()	
Default constructor.	
ActuationDataType(int type, string IoMTData)	
~ActuationDataType()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of actuation data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	Actuation data from another MThing shall follow `MTDL` and `MACV` defined in subclauses 5 and 7.
Public Member Functions	
Modifier and Type	**Public Member Functions and Description**
int	getType()
void	setType(int type)

string	getIoMTData()
void	setIoMTData(string IoMTData)

4.8.6.4 C structure

Table 126 presents return class types of IoMT actuation data in C.

Table 126 — Return class types of IoMT actuation data in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of actuation data. 0 = None, 1 = XML, 2 = Binary
Char*	IoMTData
	Actuation data from another MThing shall follow `MTDL` and `MACV` defined in subclauses 5 and 7.

4.8.7 IoMT AnalysedDataType

4.8.7.1 General

This subclause contains the analysed data description class types of IoMT analysers following the data format specification in subclauses 5 and 8.

Table 127 presents return class types of IoMT analysed data in JAVA.

Table 127 — Return class types of IoMT analysed data in JAVA

Constructor
Constructor and Description
AnalysedDataType()
Default constructor.

AnalysedDataType(int type, string IoMTData)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of analysed data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	It is analysed data of an IoMT analyser that shall follow `MTDL` and `MAOV` defined in subclauses 5 and 8.
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)
string	getIoMTData()
void	setIoMTData(string IoMTData)

4.8.7.2 C++ class

Table 128 presents return class types of IoMT analysed data in C++.

Table 128 — Return class types of IoMT analysed data in C++

Constructor and Destructor	
Constructor and Destructor Description	
AnalysedDataType()	
Default constructor.	
AnalysedDataType(int type, string IoMTData)	
~AnalysedDataType()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of analysed data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	It is analysed data of an IoMT analyser that shall follow the specifications of MTDI and MAOV defined in subclauses 5 and 8.
Public Member Functions	
Modifier and Type	**Public Member Functions and Description**
int	getType()
void	setType(int type)

string	getIoMTData()
void	setIoMTData(string IoMTData)

4.8.7.3 C structure

Table 129 presents return class types of IoMT analysed data in C.

Table 129 — Return class types of IoMT analysed data in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of analysed data. 0 = None, 1 = XML, 2 = Binary
Char*	IoMTData
	It is analysed data of an IoMT analyser that shall follow the specifications of `MTDL` and `MAOV` defined in subclauses 5 and 8.

4.8.8 IoMT CapabilityListType

4.8.8.1 General

This subclause contains the capability data description class types of MThings following the data format specification in A.2.

4.8.8.2 JAVA class

Table 130 presents return class types of IoMT capability list data in JAVA.

Table 130 — Return class types of IoMT capability list data in JAVA

Constructor	
Constructor and Description	
CapabilityListType()	
Default constructor.	
CapabilityListType(int type, string IoMTData)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of MThing capability list data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	It is a capability list of an MThing.
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)
string	getIoMTData()
void	setIoMTData(string IoMTData)

4.8.8.3 C++ class

Table 131 presents return class types of IoMT capability list data in C++.

Table 131 — Return class types of IoMT capability list data in C++

Constructor and Destructor	
Constructor and Destructor Description	
CapabilityListType()	
Default constructor.	
CapabilityListType(int type, string IoMTData)	
~CapabilityListType()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of MThing capability list data. 0 = None, 1 = XML, 2 = Binary
string	IoMTData
	It is a capability list of an MThing.
Public Member Functions	
Modifier and Type	**Public Member Functions and Description**
int	getType()
Void	setType(int type)

string	getIoMTData()
Void	setIoMTData(string IoMTData)

4.8.8.4 C structure

Table 132 presents return class types of IoMT capability list data in C.

Table 132 — Return class types of IoMT capability list data in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of MThing capability list data. 0 = None, 1 = XML, 2 = Binary
char*	IoMTData
	It is a capability list of an MThing.

4.8.9 IoMT MThingInfoType

4.8.9.1 General

This subclause contains the MThing information description class types of MThings following the data format specification in subclause 5.5.

4.8.9.2 JAVA class

Table 133 presents return class types of MThing information in JAVA.

Table 133 — Return class types of MThing information in JAVA

Constructor	
Constructor and Description	
MThingInfoType()	
Default constructor.	
MThingInfoType(int type, string MThingInfo)	
Fields	
Modifier and Type	**Field and Description**
int	type
	It is a returning type of MThing information. 0 = None, 1 = XML, 2 = Binary
string	MThingInfo
	MThing information that the data format of `MThingInfo` shall represent specified in subclause 5.5.
Methods	
Modifier and Type	**Method and Description**
int	getType()
void	setType(int type)
string	getMThingInfo()
void	setMThingInfo(string MThingInfo)

4.8.9.3 C++ class

Table 134 presents return class types of MThing information in C++.

Table 134 — Return class types of MThing information in C++

Constructor and Destructor	
Constructor and Destructor Description	
MThingInfoType()	
Default constructor.	
MThingInfoType(int type, string MThingInfo)	
~ MThingInfoType()	
Default destructor.	
Member Data	
Modifier and Type	**Member Data and Description**
int	type
	It is a returning type of MThing information. 0 = None, 1 = XML, 2 = Binary
string	MThingInfo
	MThing information that the data format of `MThingInfo` shall represent specified in subclause 5.5.
Public Member Functions	
Modifier and Type	**Public Member Functions and Description**
int	getType()
void	setType(int type)

string	getMThingInfo()
void	setMThingInfo(string MThingInfo)

4.8.9.4 C structure

Table 135 presents return class types of MThing information in C.

Table 135 — Return class types of MThing information in C

Data Field	
Modifier and Type	**Data Field and Description**
int	type
	It is a returning type of MThing information. 0 = None, 1 = XML, 2 = Binary
char*	MThingInfo
	MThing information that the data format of `MThingInfo` shall represent specified in subclause 5.5.

5 Media thing description language

5.1 General

This subclause describes a basic structure of the tools in this document in the form of media thing description language (MTDL), including the schema wrapper conventions, basic data types, root elements, and top-level elements.

5.2 Schema wrapper

The syntax of description tools specified in this subclause is provided as a collection of schema components, including type definitions and element declarations. To form a valid schema document, users can gather these schema components in the same document with the following declaration defining, in particular, the target namespace and the namespaces prefixes.

```
<?xml version="1.0"?>
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mtdl="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
  xmlns:mpegvct="urn:mpeg:mpeg-v:2017:01-CT-NS"
  targetNamespace="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="ISO/IEC 23093-3" id="MPEG-IOMT-MTDL.xsd">

  <import namespace="urn:mpeg:mpeg7:schema:2004"
schemaLocation="http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
7_schema_files/mpeg7-v2.xsd"/>
  <import namespace="urn:mpeg:mpeg-v:2017:01-CT-NS"
schemaLocation="https://standards.iso.org/iso-iec/23005/-6/ed-4/en/MPEG-V-
CT.xsd"/>
```

The following line should be appended to the resulting schema document to obtain a well-formed XML document.

```
</schema>
```

5.3 Mnemonics for binary representations

The following mnemonics shall be used as defined in ISO/IEC 15938-3 to describe different data types used in the definitions of binary representations defined by this document.

Mnemonics	Description
bslbf	A bit string, left bit first, where "left" is the order in which bits are written in ISO/IEC 23093. Bit strings are generally written as a string of 1s and 0s within single quote marks, e.g., '1000 0001'. Blanks within a bit string are for ease of reading and have no significance. For convenience, large strings are occasionally written in hexadecimal, in which case conversion to a binary in a conventional manner will yield the value of the bit string. Thus, the left-most hexadecimal digit is first, and in each hexadecimal digit, the most significant of the four digits is first.
vluimsbf5	Variable length unsigned integer most significant bit first representation consisting of two parts. The first part defines the number n of 4-bit bit fields used for the value representation, encoded by a sequence of n-1 "1" bits, followed by a"0" bit signalling its end. The second part contains the value of the integer encoded using the number of bit fields specified in the first part.
uimsbf	Unsigned integer, most significant bit first.
simsbf	Signed integer, in tow's complement format, most significant bit (sign) first.

Mnemonics	Description
fsfb	Float (32 bit), sign bit first. The semantics of the bits within the float is specified in the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std. 754-1985 [2]).
UTF-8	Binary string encoding defined in ISO 10646 [3]/IETF RFC 2279 [4], preceded by its size in bytes coded as vluimsbf5.

5.4 Base data types and elements

5.4.1 General

This subclause specifies base data types and elements of MThing data formats. An MThing can have a data-id, a device id, IP address, and a URL as base attributes. Besides, data can include the supported cryptocurrencies and legal tenders to use (or borrow) functionalities, resources, and information of MThings.

5.4.2 Syntax

```
<attributeGroup name="MThingBaseAttributes">
    <attribute name="id" type="string" use="optional"/>
    <attribute name="idRef" type="string" use="required"/>
    <attribute name="ipAddress" type="string" use="optional"/>
    <attribute name="url" type="anyURI" use="optional"/>
</attributeGroup>

<element name="MSensor" type="mtdl:MSensorType"/>
<element name="MActuator" type="mtdl:MActuatorType"/>
<element name="MAnalyser" type="mtdl:MAnalyserType"/>
<element name="MStorage" type="mtdl:MStorageType"/>
<element name="MManager" type="mtdl:MManagerType"/>
<element name="MAggregator" type="mtdl:MAggregatorType"/>

<complexType name="SupportedTokenListType">
       <sequence minOccurs="1" maxOccurs="1">
                <element name="cryptocurrencyList"
type="mtdl:cryptocurrencyListType" minOccurs="0" maxOccurs="1"/>
                <element name="legalTenderList" type="mtdl:LegalTenderListType"
minOccurs="0" maxOccurs="1"/>
       </sequence>
</complexType>

<complexType name="cryptocurrencyListType">
    <sequence>
        <element name="cryptocurrency" type="mtdl:cryptocurrencyType"
minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="cryptocurrencyType">
    <attribute name="cryptocurrencyName" type="mpeg7:termReferenceType"
use="required"/>
    <attribute name="cryptocurrencyWalletAddress" type="string" use="required"/>
</complexType>
```

```
<complexType name="LegalTenderListType">
    <sequence>
        <element name="legalTender" type="mtdl:LegalTenderType" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="LegalTenderType">
    <attribute name="legalTenderName" type="mpeg7:termReferenceType"
use="required"/>
    <attribute name="legalTenderWalletAddress" type="string" use="required"/>
</complexType>
```

5.4.3 Binary Representation

MThingBaseAttributesType {	**Number of Bits**	**Mnemonic**
idFlag	1	bslbf
ipAddressFlag	1	bslbf
urlFlag	1	bslbf
if(idFlag) {		
id	See ISO 10646	UTF-8
}		
idRef	See ISO 10646	UTF-8
if(ipAddressFlag) {		
ipVersionFlag	1	bslbf
If(ipVersionFlag)		
ipAddress	128	bslbf
else		
ipAddress	32	bslbf
}		
if(urlFlag) {		
url	See ISO 10646	UTF-8
}		
}		

MSensor {	**Number of Bits**	**Mnemonic**
MSensorType		MSensorType
}		

MActuatorType {	**Number of Bits**	**Mnemonic**
MActuatorType		MActuatorType
}		

MAnalyserType {	**Number of Bits**	**Mnemonic**
MAnalyserType		MAnalyserType
}		

MStorageType {	**Number of Bits**	**Mnemonic**
MStorageType		MStorageType
}		

MManagerType {	**Number of Bits**	**Mnemonic**
MManagerType		MManagerType
}		

MAggregatorType {	**Number of Bits**	**Mnemonic**
MAggregatorType		MAggregatorType
}		

SupportedTokenListType {	**Number of Bits**	**Mnemonic**
cryptocurrencyListFlag	1	bslbf
legalTenderListFlag	1	bslbf
if(cryptocurrencyListFlag) {		
cryptocurrencyList		cryptocurrencyListType
}		
if(legalTenderListFlag) {		
legalTenderList		LegalTenderListType
}		

cryptocurrencyListType {	**Number of Bits**	**Mnemonic**
cryptocurrencyFlag	1	bslbf
if(cryptocurrencyFlag) {		
NumOfCryptocurrency		vluimsbf5
for(k=0;k<NumOfCryptocurrency;k++) {		
cryptocurrency[k]		cryptocurrencyType
}		
}		
}		

cryptocurrencyType {	**Number of Bits**	**Mnemonic**
cryptocurrencyName	8	bslbf
cryptocurrencyWalletAddressLength		vluimsbf5
cryptocurrencyWalletAddress	cryptocurrencyWalletAddressLength*8	bslbf
}		

LegalTenderListType {	**Number of Bits**	**Mnemonic**
legalTenderFlag	1	bslbf
if(legalTenderFlag) {		
NumOfLegalTender		vluimsbf5
for(k=0;k<NumOfLegalTender;k++) {		
legalTender[k]		cryptocurrencyType
}		
}		
}		

LegalTenderType {	**Number of Bits**	**Mnemonic**
LegalTenderName	8	bslbf
LegalTenderWalletAddressLength		vluimsbf5
LegalTenderWalletAddress	LegalTenderWalletAddressLength*8	bslbf
}		

5.4.4 Semantics

Name	**Definition**
`idFlag`	This field, which is only present in the binary representation, indicates the presence of the `id` attribute. If it is set to "1," the `id` attribute follows.
`ipAddressFlag`	This field, which is only present in the binary representation, indicates the presence of the `ipAddress` attribute. If it is set to "1," the `ipAddress` attribute follows.
`urlFlag`	This field, which is only present in the binary representation, indicates the presence of `url` attribute. If it is set to "1," the `url` attribute follows.
`id`	It describes the unique identifier of this specific data instance.
`idRef`	It describes the unique identifier of the MThing that this description is bound to.

Name	Definition
ipVersionFlag	This field, which is only present in the binary representation, indicates the version of the ipAddress attribute. If it is set to "1," the ipAddress follows the IPv6 protocol; otherwise, the ipAddress follows the IPv4 protocol.
ipAddress	It describes the IP address of the MThing that this description is bound to.
url	It describes a URL address of the MThing that this description is bound to.

Name	Definition
MSensor	It describes an MSensor.
MActuator	It describes a MActuator.
MAnalyser	It describes a MAnalyser.
MStorage	It describes an MStorage.
MManager	It describes an MManager.
MAggregator	It describes a MAggregator.

Name	Definition
SupportedTokenListType	Tool for describing a media token (MToken) list which an MThing supports.
cryptocurrencyListFlag	This field, which is only present in the binary representation, indicates the presence of the cryptocurrencyList element. If it is set to "1," the cryptocurrencyList element follows.
legalTenderListFlag	This field, which is only present in the binary representation, indicates the presence of the legalTenderList element. If it is set to "1," the legalTenderList element follows.
cryptocurrencyList	It describes a list of cryptocurrencies that MThing supports.
legalTenderList	It describes a list of legal tenders that MThing supports.

Name	Definition
cryptocurrencyListType	Tool for describing a cryptocurrency list that an MThing supports.
cryptocurrencyFlag	This field, which is only present in the binary representation, indicates the presence of the cryptocurrency element. If it is set to "1," the cryptocurrency element follows.
NumOfCryptocurrency	This field, which is only present in the binary representation, specifies the number of the cryptocurrency.
cryptocurrency	It describes the cryptocurrency which is used for a transaction.

Name	Definition
cryptocurrencyType	Tool for describing a cryptocurrency.
cryptocurrencyName	It describes the alphabetic code of the cryptocurrency. The type of name shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the cryptocurrencyCS defined in A.5.1
cryptocurrencyWalletAddressLegnth	This field, which is only present in the binary representation, describes the length of the cryptocurrencyWalletAddress attribute in bytes.
cryptocurrencyWalletAddress	It describes the wallet address of the cryptocurrency.

Name	Definition
LegalTenderListType	Tool for describing a list of legal tenders.
legalTenderFlag	This field, which is only present in the binary representation, indicates the presence of the legalTender element. If it is set to "1," the legalTender attribute follows.
NumOfLegalTender	This field, which is only present in the binary representation, specifies the number of the legalTender.
legalTender	It describes a legal tender that is used for a transaction.

Name	Definition
LegalTenderType	Tool for describing a legal tender.
legalTenderName	It describes the alphabetic code of the legal tender. The type of name shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the LegalTenderCS defined in A.5.2 (Table A.22).
legalTenderWalletAddressLength	This field, which is only present in the binary representation, describes the length of the legalTenderWalletAddress attribute in bytes.
legalTenderWalletAddress	It describes the wallet address of the legal tender.

5.5 Root element

5.5.1 General

This subclause specifies a root element that describes the data (i.e., MThing information) from an MThing. The root element can include a timestamp, a supported token (cryptographies and legal tenders) list, the type of the MThing, and base attributes. Data formed by the root element can be delivered or exchanged between MThings for further MThing collaborative services.

5.5.2 Syntax

```
<element name="MThingInfo">
    <complexType>
        <sequence>
            <element name="TimeStamp" type="mpegvct:AbsoluteTimeType"
minOccurs="0"/>
            <element name="supportedTokenList" type="mtdl:SupportedTokenListType"
minOccurs="0" maxOccurs="1"/>
            <choice>
                <element ref="mtdl:MSensor"/>
                <element ref="mtdl:MActuator"/>
                <element ref="mtdl:MAnalyser"/>
                <element ref="mtdl:MStorage"/>
                <element ref="mtdl:MManager"/>
                <element ref="mtdl:MAggregator"/>
            </choice>
        </sequence>
        <attributeGroup ref="mtdl:MThingBaseAttributes"/>
    </complexType>
</element>
```

5.5.3 Binary Representation

MThingInfo {	Number of Bits	Mnemonic
TimeStampFlag	1	bslbf
supportedTokenListFlag	1	bslbf
MThingType	4	bslbf
if(TimeStampFlag) {		
TimeStamp		mpegvct:AbsoluteTimeType
}		
if(supportedTokenListFlag) {		
supportedTokenList		supportedTokenListType
}		
if(MThingType==0000) {		
MSensor		MSensorType
}		
else if(MThingType==0001) {		
MActuator		MActuatorType
}		
else if(MThingType==0010) {		
MAnalyser		MAnalyserType
}		
else if(MThingType==0011) {		
MStorage		MStorageType
}		
else if(MThingType==0100) {		
MManager		MManager
}		
else if(MThingType==0101) {		
MAggregator		MAggregatorType
}		
MThingBaseAttributes		MThingBaseAttributesType
}		

5.5.4 Semantics

Name	Definition
`MThingInfo`	It serves as the root element for MThing metadata.
`TimeStampFlag`	This field, which is only present in the binary representation, indicates the presence of the `TimeStamp` element. If it is set to "1," the `TimeStamp` element follows.
`supportedTokenListFlag`	This field, which is only present in the binary representation, indicates the presence of the `supportedTokenList` element. If it is set to "1," the `supportedTokenList` element follows.

Name	Definition
MThingType	This field, which is only present in the binary representation, describes which MThing type shall be used. In the binary description, the mapping between terms of MThings and its 4bit representation is used as follows: - MSensor: 0000, - MActuator: 0001, - MAnalyser: 0010, - MStorage: 0011, - MManager: 0100, - MAggregator: 0101, - Reserved: 0110-1111.
timeStamp	It provides the absolute time information of MThing metadata as defined in ISO/IEC 23005-6.
supportedTokenList	It describes a list of tokens that MThing supports.

5.6 Media sensor description language

5.6.1 General

This subclause specifies tools for describing media sensors. The following subclause defines a complex type of MSensorType, which contains the MSensor capability description. And there is an abstract type of SensedDataBaseType, which the sensed data of an individual MSensor should inherit.

5.6.2 Syntax

```
<!-- ################################################ -->
<!--  MSensor Base Type                               -->
<!-- ################################################ -->
<complexType name="MSensorType">
    <sequence>
        <element name="sensorCapabilityList" type="mtdl:SensorCapabilityListType"
minOccurs="0"/>
        <element name="sensedData" type="mtdl:SensedDataBaseType" minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="SensorCapabilityListType">
    <sequence>
        <element name="sensorCapability" type="mtdl:SensorCapabilityType"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="listType" type="mtdl:sensorCapabilityListEnumType"/>
</complexType>
```

```
<simpleType name="sensorCapabilityListEnumType">
    <restriction base="string">
        <enumeration value="allSensorCapabilityList"/>
        <enumeration value="availableSensorCapabilityList"/>
        <enumeration value="appliedSensorCapabilityList"/>
    </restriction>
</simpleType>

<complexType name="SensorCapabilityType">
    <sequence>
        <element name="sensorCapabilityParameter"
type="mtdl:SensorCapabilityParameterType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="sensorCapabilityName" type="mpeg7:termReferenceType"
use="required"/>
</complexType>

<complexType name="SensorCapabilityParameterType">
    <attribute name="sensorCapabilityParameterName"
type="mpeg7:termReferenceType" use="required"/>
</complexType>

<complexType name="SensedDataBaseType" abstract="true"/>
```

5.6.3 Binary Representation

MSensorType {	**Number of Bits**	**Mnemonic**
sensorCapabilityListFlag	1	bslbf
sensedDataFlag	1	bslbf
If(sensorCapabilityListFlag) {		
sensorCapabilityList		SensorCapabilityListType
}		
if(sensedDataFlag) {		
sensedData		SensedDataBaseType
}		
}		

SensorCapabilityListType {	**Number of Bits**	**Mnemonic**
listTypeFlag	1	bslbf
NumOfSensorCapability		vluimsbf5
if(k=0;k<NumOfSensorCapability;k++) {		
sensorCapability[k]		SensorCapabilityType
}		
if(listTypeFlag) {		
listType	2	bslbf
}		
}		

SensorCapabilityType {	Number of Bits	Mnemonic
sensorCapabilityParameterFlag	1	bslbf
if(sensorCapabilityParameterFlag) {		
NumOfSensorCapabilityParameter		vluimsbf5
if(k=0;k<NumOfSensorCapabilityParameter;k++) {		
sensorCapabilityParameter[k]		SensorCapabilityParameterType
}		
}		
sensorCapabilityName	8	bslbf
}		

SensorCapabilityParameterType {	Number of Bits	Mnemonic
sensorCapabilityParameterName	8	bslbf
}		

SensedDataBaseType {	Number of Bits	Mnemonic
}		

5.6.4 Semantics

Name	Definition
`sensorCapabilityListFlag`	This field, which is only present in the binary representation, indicates the presence of the `sensorCapabilityList` element. If it is set to "1," the `sensorCapabilityList` element follows.
`sensedDataFlag`	This field, which is only present in the binary representation, indicates the presence of the `sensedData` element. If it is set to "1," the `sensedData` element follows.
`sensorCapabilityList`	List of the MSensor's capabilities.
`sensedData`	Sensed data from an MSensor.
`SensorCapabilityListType`	Tool for describing sensor capability.
`listTypeFlag`	This field, which is only present in the binary representation, indicates the presence of the `listType` attribute. If it is set to "1," the `listType` attribute follows.
`NumOfSensorCapability`	This field, which is only present in the binary representation, specifies the number of `sensorCapability`.
`sensorCapability`	List of the MSensor's full capabilities.

Name	Definition
listType	Tool for specify the type of sensor capabilities among allSensorCapabilityList, availableSensorCapabilityList, and appliedSensorCapabilityList. In the binary description, the mapping is used between the listType and the 2bit binary representation as follows: - allSensorCapabilityList: 00, - availableSensorCapabilityList: 01, - appliedSensorCapabilityList: 10, - Reserved: 11.
sensorCapabilityListEnumType	It is an enumeration list that can be one of the capability types.
SensorCapabilityType	Tool for describing capabilities.
sensorCapabilityParameterFlag	This field, which is only present in the binary representation, indicates the presence of the sensorCapabilityParameter element. If it is set to "1," the sensorCapabilityParameter element follows.
NumOfSensorCapabilityParameter	This field, which is only present in the binary representation, specifies the number of sensorCapabilityParameter.
sensorCapabilityParameter	List of the MSensor's capability parameters.
sensorCapabilityName	Name of a capability of MSensor. The type of capability shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the SensorCapabilityCS defined in A.2.1.
SensorCapabilityParameterType	Tool for describing capability parameters.
sensorCapabilityParameterName	Name of a capability parameter. The type of the capability parameter shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the SensorCapabilityParameterCS defined in A.3.1 (Table A.8).
SensedDataBaseType	Tool for describing sensed data. SensedDataBaseType shall be instantiated with one of the sensed data defined in Subclause 6.

5.6.5 Example

This example shows a MSensor with a data instance ID of "MS001-0012" and a device ID of "MS001." This MSensor has capabilities of "SENSOR_CAPTURE_VIDEO" and "SENSOR_CAPTURE_AUDIO."

```
<mtdl:MThingInfo id="MS001-0012" idRef="MS001">
    <mtdl:MSensor>
        <mtdl:sensorCapabilityList listType="allSensorCapabilityList">
            <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:SENSOR_CAPTURE_VIDEO"/>
            <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:SENSOR_CAPTURE_AUDIO"/>
        </mtdl:sensorCapabilityList>
    </mtdl:MSensor>
</mtdl:MThingInfo>
```

5.7 Media actuator description language

5.7.1 General

This subclause specifies tools for describing media actuators. The following subclause defines a complex type of MActuatorType, which contains the MActuator capability description and an abstract type of ActuationDataBaseType, which is the actuation commands of an individual MActuator should inherit.

5.7.2 Syntax

```
<!-- ################################################ -->
<!--  MActuator Base Type                             -->
<!-- ################################################ -->
<complexType name="MActuatorType">
    <sequence>
        <element name="actuatorCapabilityList"
type="mtdl:ActuatorCapabilityListType" minOccurs="0"/>
        <element name="actuationData" type="mtdl:ActuationDataBaseType"
minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="ActuatorCapabilityListType">
    <sequence>
        <element name="actuatorCapability" type="mtdl:ActuatorCapabilityType"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="listType" type="mtdl:ActuatorCapabilityListEnumType"/>
</complexType>

<simpleType name="ActuatorCapabilityListEnumType">
    <restriction base="string">
        <enumeration value="allActuatorCapabilityList"/>
        <enumeration value="availableActuatorCapabilityList"/>
        <enumeration value="appliedActuatorCapabilityList"/>
    </restriction>
</simpleType>

<complexType name="ActuatorCapabilityType">
    <sequence>
        <element name="actuatorCapabilityParameter"
type="mtdl:ActuatorCapabilityParameterType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="actuatorCapabilityName" type="mpeg7:termReferenceType"
use="required"/>
</complexType>
```

```
<complexType name="ActuatorCapabilityParameterType">
   <attribute name="actuatorCapabilityParameterName"
type="mpeg7:termReferenceType" use="required"/>
</complexType>

<complexType name="ActuationDataBaseType" abstract="true"/>
```

5.7.3 Binary Representation

MActuatorType {	**Number of Bits**	**Mnemonic**
actuatorCapabilityListFlag	1	bslbf
actuationDataFlag	1	bslbf
If(actuatorCapabilityListFlag) {		
ActuatorCapabilityList		ActuatorCapabilityListType
}		
if(actuationDataFlag) {		
ActuationData		ActuationDataBaseType
}		
}		

ActuatorCapabilityListType {	**Number of Bits**	**Mnemonic**
listTypeFlag	1	bslbf
NumOfActuatorCapability		vluimbsf5
if(k=0;k<NumOfActuatorCapability;k++) {		
actuatorCapability[k]		ActuatorCapabilityType
}		
if(listTypeFlag) {		
listType	2	bslbf
}		
}		

ActuatorCapabilityType {	**Number of Bits**	**Mnemonic**
actuatorCapabilityParameterFlag	1	bslbf
if(actuatorCapabilityParameterFlag) {		
NumOfActuatorCapabilityParameter		vluimsbf5
if(k=0;k<NumOfActuatorCapabilityParameter;k++) {		
actuatorCapabilityParameter[k]		ActuatorCapabilityParameterType
}		
}		
actuatorCapabilityName	8	bslbf
}		

ActuatorCapabilityParameterType {	**Number of Bits**	**Mnemonic**
actuatorCapabilityParameterName	8	bslbf
}		

ActuationDataBaseType {	**Number of Bits**	**Mnemonic**
}		

5.7.4 Semantics

Name	**Definition**
`actuatorCapabilityListFlag`	This field, which is only present in the binary representation, indicates the presence of the `actuatorCapabilityList` element. If it is set to "1," the `actuatorCapabilityList` element follows.
`actuationDataFlag`	This field, which is only present in the binary representation, indicates the presence of the `actuationData` element. If it is set to "1," the `actuationData` element follows.
`actuatorCapabilityList`	List of the MActuator's capabilities.
`actuationData`	Actuation command from other MThings.
`ActuatorCapabilityListType`	Tool for describing actuator capability.
`listTypeFlag`	This field, which is only present in the binary representation, indicates the presence of the `listType` attribute. If it is set to "1," the `listType` attribute follows.
`NumOfActuatorCapability`	This field, which is only present in the binary representation, specifies the number of `actuatorCapability`.
`actuatorCapability`	List of the MActuator's full capabilities.
`listType`	Tool for specify the type of actuator capabilities among `allActuatorCapabilityList`, `availableActuatorCapabilityList`, and `appliedActuatorCapabilityList`. In the binary description, the mapping is used between the listType and the 2bit binary representation as follows: - allActuatorCapabilityList: 00, - availableActuatorCapabilityList: 01, - appliedActuatorCapabilityList: 10, - Reserved: 11.

Name	Definition
actuatorCapabilityListEnumType	It is an enumeration list that can be one of the capability types.
ActuatorCapabilityType	Tool for describing capabilities.
actuatorCapabilityParameterFlag	This field, which is only present in the binary representation, indicates the presence of the actuatorCapabilityParameter element. If it is set to "1," the actuatorCapabilityParameter element follows.
NumOfActuatorCapabilityParameter	This field, which is only present in the binary representation, specifies the number of actuatorCapabilityParameter
actuatorCapabilityParameter	List of the MActuator's capability parameters.
actuatorCapabilityName	Name of a capability of MActuator. The type of capability shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the ActuatorCapabilityCS defined in A.2.2.
ActuatorCapabilityParameterType	Tool for describing capability parameters.
actuatorCapabilityParameterName	Name of a capability parameter. The type of the capability parameter shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the ActuatorCapabilityParameterCS defined in A.3.2.
ActuationDataBaseType	Tool for describing actuation data. ActuationDataBaseType shall be instantiated with one of the actuation data defined in Subclause 7.

5.7.5 Example

This example shows a MActuator with a data instance ID of "MAC003-0001" and a device ID of "MAC003." This MActuator has capabilities of "ACTUATOR_SPRAY_WATER," "ACTUATOR_SPRAY_FOG," and "ACTUATOR_SPRAY_BUBBLE."

```
<mtdl:MThingInfo id="MAC003-0001" idRef="MAC003">
    <mtdl:MActuator>
        <mtdl:actuatorCapabilityList listType="allActuatorCapabilityList">
            <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
ActuatorCapabilityCS-NS:ACTUATOR_SPRAY_WATER"/>
            <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
ActuatorCapabilityCS-NS:ACTUATOR_SPRAY_FOG"/>
            <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
ActuatorCapabilityCS-NS:ACTUATOR_SPRAY_BUBBLE"/>
        </mtdl:actuatorCapabilityList>
    </mtdl:MActuator>
</mtdl:MThingInfo>
```

5.8 Media analyser description language

5.8.1 General

This subclause specifies tools for describing media analysers. The following subclause defines a complex type of `MAnalyserType`, which contains the MAnalyser capability description, and an abstract type of AnalysedDataBaseType, which is the analysed data of an individual MAnalyser should inherit.

5.8.2 Syntax

```
<!-- ################################################ -->
<!--  MAnalyser Base Type                             -->
<!-- ################################################ -->
<complexType name="MAnalyserType">
    <sequence>
        <element name="analyserCapabilityList"
type="mtdl:AnalyserCapabilityListType" minOccurs="0"/>
        <element name="analysedData" type="mtdl:AnalysedDataBaseType"
minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="AnalyserCapabilityListType">
    <sequence>
        <element name="analyserCapability" type="mtdl:AnalyserCapabilityType"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="listType" type="mtdl:AnalyserCapabilityListEnumType"/>
</complexType>

<simpleType name="AnalyserCapabilityListEnumType">
    <restriction base="string">
        <enumeration value="allAnalyserCapabilityList"/>
        <enumeration value="availableAnalyserCapabilityList"/>
        <enumeration value="appliedAnalyserCapabilityList"/>
    </restriction>
</simpleType>

<complexType name="AnalyserCapabilityType">
    <sequence>
        <element name="analyserCapabilityParameter"
type="mtdl:AnalyserCapabilityParameterType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="analyserCapabilityName" type="mpeg7:termReferenceType"
use="required"/>
</complexType>

<complexType name="AnalyserCapabilityParameterType">
    <attribute name="analyserCapabilityParameterName"
type="mpeg7:termReferenceType" use="required"/>
</complexType>

<complexType name="AnalysedDataBaseType" abstract="true"/>
```

5.8.3 Binary Representation

MAnalyserType {	Number of Bits	Mnemonic
analyserCapabilityListFlag	1	bslbf
analysedDataFlag	1	bslbf
if(analyserCapabilityListFlag) {		
analyserCapabilityList		AnalyserCapabilityListType
}		
if(analysedDataFlag) {		
analysedData		AnalysedDataBaseType
}		
}		

AnalyserCapabilityListType{	Number of Bits	Mnemonic
listTypeFlag	1	bslbf
NumOfAnalyserCapability	5+	vluimbsf5
if(k=0;k<NumOfAnalyserCapability;k++) {		
analyserCapability[k]		AnalyserCapabilityType
}		
if(listTypeFlag) {		
listType	2	bslbf
}		
}		

AnalyserCapabilityType{	Number of Bits	Mnemonic
analyserCapabilityParameterFlag	1	bslbf
if(analyserCapabilityParameterFlag) {		
NumOfAnalyserCapabilityParameter	5+	vluimsbf5
if(k=0;k<NumOfAnalyserCapabilityParameter;k++) {		
analyserCapabilityParameter[k]		AnalyserCapabilityParameterType
}		
}		
analyserCapabilityName	8	bslbf
}		

AnalyserCapabilityParameterType {	Number of Bits	Mnemonic
analyserCapabilityParameterName	8	bslbf
}		

AnalysedDataBaseType {	**Number of Bits**	**Mnemonic**
}		

5.8.4 Semantics

Name	**Definition**
analyserCapabilityListFlag	This field, which is only present in the binary representation, indicates the presence of the analyserCapabilityList element. If it is set to "1," the analyserCapabilityList element follows.
analysedDataFlag	This field, which is only present in the binary representation, indicates the presence of the analysedData element. If it is set to "1," the analysedData element follows.
analyserCapabilityList	List of the MAnalyser's capabilities.
analysedData	Analysed data from a MAnalyser.
AnalyserCapabilityListType	Tool for describing analyser capability.
listTypeFlag	This field, which is only present in the binary representation, indicates the presence of the listType attribute. If it is set to "1," the listType attribute follows.
NumOfAnalyserCapability	This field, which is only present in the binary representation, specifies the number of analyserCapability.
analyserCapability	List of the MAnalyser's full capabilities.
listType	Tool for specifying the type of analyser capabilities among allAnalyserCapabilityList, availableAnalyserCapabilityList, and appliedAnalyserCapabilityList. In the binary description, the mapping is used between the listType and the 2bit binary representation as follows: - allAnalyserCapabilityList: 00, - availableAnalyserCapabilityList: 01, - appliedAnalyserCapabilityList: 10, - Reserved: 11.
analyserCapabilityListEnumType	It is an enumeration list that can be one of the capability types.
AnalyserCapabilityType	Tool for describing capabilities.

Name	Definition
analyserCapabilityParameterFlag	This field, which is only present in the binary representation, indicates the presence of the analyserCapabilityParameter element. If it is set to "1," the analyserCapabilityParameter element follows.
NumOfAnalyserCapabilityParameter	This field, which is only present in the binary representation, specifies the number of analyserCapabilityParameter
analyserCapabilityParameter	List of the MAnalyser's capability parameters.
analyserCapabilityName	Name of a capability of MAnalyser. The type of capability shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the AnalyserCapabilityCS defined in A.2.3.
AnalyserCapabilityParameterType	Tool for describing capability parameters.
analyserCapabilityParameterName	Name of a capability parameter. The type of the capability parameter shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the AnalyserCapabilityParameterCS defined in A.3.3 (Table A.10).
AnalysedDataBaseType	Tool for describing analysed data. AnalysedDataBaseType shall be instantiated with one of the analysed data defined in Subclause 8.

5.8.5 Example

This example shows a MAnalyser with a data instance ID of "MAZ005-0001" and a device ID of "MAZ005." This MAnalyser has a capability of "ANALYSER_GUIDE_DIRECTION."

```
<mtdl:MThingInfo id="MAZ005-0001" idRef="MAZ005">
    <mtdl:MAnalyser>
        <mtdl:analyserCapabilityList listType="allAnalyserCapabilityList">
            <mtdl:analyserCapability analyserCapabilityName="urn:mpeg:mpeg-IoMT:01-
AnalyserCapabilityCS-NS:ANALYSER_GUIDE_DIRECTION"/>
        </mtdl:analyserCapabilityList>
    </mtdl:MAnalyser>
</mtdl:MThingInfo>
```

5.9 Media storage description language

5.9.1 General

This subclause specifies tools for describing media storage. The following subclause defines a complex type of MStorageType, which contains the MStorage capability description. And there is an abstract type of StorageDataBaseType, which the storage commands of an individual MStorage should inherit.

5.9.2 Syntax

```
<!-- ################################################ -->
<!--  MStorage Base Type                              -->
<!-- ################################################ -->
<complexType name="MStorageType">
    <sequence>
        <element name="storageCapabilityList"
type="mtdl:StorageCapabilityListType" minOccurs="0"/>
        <element name="storageCommand" type="mtdl:StorageCommandBaseType"
minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="StorageCapabilityListType">
    <sequence>
        <element name="storageCapability" type="mtdl:StorageCapabilityType"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="listType" type="mtdl:StorageCapabilityListEnumType"/>
</complexType>

<simpleType name="StorageCapabilityListEnumType">
    <restriction base="string">
        <enumeration value="allStorageCapabilityList"/>
        <enumeration value="availableStorageCapabilityList"/>
        <enumeration value="appliedStorageCapabilityList"/>
    </restriction>
</simpleType>

<complexType name="StorageCapabilityType">
    <sequence>
        <element name="storageCapabilityParameter"
type="mtdl:StorageCapabilityParameterType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="storageCapabilityName" type="mpeg7:termReferenceType"
use="required"/>
</complexType>

<complexType name="StorageCapabilityParameterType">
    <attribute name="storageCapabilityParameterName"
type="mpeg7:termReferenceType" use="required"/>
</complexType>

<complexType name="StorageCommandBaseType" abstract="true"/>
```


5.9.3 Binary Representation

MStorageType {	**Number of Bits**	**Mnemonic**
storageCapabilityListFlag	1	bslbf
storageCommandFlag	1	bslbf
if(storageCapabilityListFlag) {		
storageCapabilityList		StorageCapabilityListType
}		

MStorageType {	Number of Bits	Mnemonic
if(storageCommandFlag) {		
storageCommand		StorageCommandBaseType
}		
}		

StorageCapabilityListType {	Number of Bits	Mnemonic
listTypeFlag	1	bslbf
NumOfStorageCapability		vluimbsf5
if(k=0;k<NumOfStorageCapability;k++) {		
storageCapability[k]		StorageCapabilityType
}		
if(listTypeFlag) {		
listType	2	bslbf
}		
}		

StorageCapabilityType {	Number of Bits	Mnemonic
storageCapabilityParameterFlag	1	bslbf
if(storageCapabilityParameterFlag) {		
NumOfStorageCapabilityParameter		vluimsbf5
if(k=0;k<NumOfStorageCapabilityParameter;k++) {		
storageCapabilityParameter[k]		StorageCapabilityParameterType
}		
}		
storageCapabilityName	8	bslbf
}		

StorageCapabilityParameterType {	Number of Bits	Mnemonic
storageCapabilityParameterName	8	bslbf
}		

StorageCommandBaseType {	Number of Bits	Mnemonic
}		

5.9.4 Semantics

Name	Definition
storageCapabilityListFlag	This field, which is only present in the binary representation, indicates the presence of the storageCapabilityList element. If it is set to "1," the storageCapabilityList element follows.
storageCommandFlag	This field, which is only present in the binary representation, indicates the presence of the storageCommand element. If it is set to "1," the storageCommand element follows.
storageCapabilityList	List of MStorage's capabilities.
StorageCapabilityListType	Tool for describing storage capability.
storageCommand	Command to control the storage.
listTypeFlag	This field, which is only present in the binary representation, indicates the presence of the listType attribute. If it is set to "1," the listType attribute follows.
NumOfStorageCapability	This field, which is only present in the binary representation, specifies the number of storageCapability.
storageCapability	List of MStorage's full capabilities.
listType	Tool for specify the type of storage capabilities among allStorageCapabilityList, availableStorageCapabilityList, and appliedStorageCapabilityList. In the binary description, the mapping is used between the listType and the 2bit binary representation as follows: - allStorageCapabilityList: 00, - availableStorageCapabilityList: 01, - appliedStorageCapabilityList: 10, - Reserved: 11.
storageCapabilityListEnumType	It is an enumeration list that can be one of the capability types.
StorageCapabilityType	Tool for describing capabilities.
storageCapabilityParameterFlag	This field, which is only present in the binary representation, indicates the presence of the storageCapabilityParameter element. If it is set to "1," the storageCapabilityParameter element follows.
NumOfStorageCapabilityParameter	This field, which is only present in the binary representation, specifies the number of storageCapabilityParameter.
storageCapabilityParameter	List of the MStorage's capability parameters.

Name	Definition
storageCapabilityName	Name of a capability of MStorage. The type of capability shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the StorageCapabilityCS defined in A.2.4.
StorageCapabilityParameterType	Tool for describing storage capability parameters.
storageCapabilityParameterName	Name of a capability parameter. The type of the capability parameter shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the StorageCapabilityParameterCS defined in A.3.4 (Table A.11).
StorageCommandBaseType	Tool for describing storage control commands. StorageCommandBaseType shall be instantiated with one of the Storage command data. (placeholder)

5.9.5 Example

This example shows an MStorage with a data instance ID of "MST006-0001" and a device ID of "MST006." This MStorage has capabilities of "STORAGE_SAVE," "STORAGE_READ," "STORAGE_DELETE," and "STORAGE_UPDATE."

```
<mtdl:MThingInfo id="MST006-0001" idRef="MST006">
    <mtdl:MStorage>
        <mtdl:storageCapabilityList listType="allStorageCapabilityList">
            <mtdl:storageCapability storageCapabilityName="urn:mpeg:mpeg-IoMT:01-
StorageCapabilityCS-NS:STORAGE_SAVE"/>
            <mtdl:storageCapability storageCapabilityName="urn:mpeg:mpeg-IoMT:01-
StorageCapabilityCS-NS:STORAGE_READ"/>
            <mtdl:storageCapability storageCapabilityName="urn:mpeg:mpeg-IoMT:01-
StorageCapabilityCS-NS:STORAGE_DELETE"/>
            <mtdl:storageCapability storageCapabilityName="urn:mpeg:mpeg-IoMT:01-
StorageCapabilityCS-NS:STORAGE_UPDATE"/>
        </mtdl:storageCapabilityList>
    </mtdl:MStorage>
</mtdl:MThingInfo>
```


5.10 Media manager description language

5.10.1 General

This subclause specifies tools for describing media thing managers, which can act as an IoMT server. This server can manage (e.g., register or unregister) the list of MThings in the network for providing IoMT services. The following subclause defines a complex type of MManagerType, which contains the MManager capability description and an abstract type of ManagerDataBaseType. The server processing data of an individual MManager should inherit.

5.10.2 Syntax

```
<!-- ################################################ -->
<!--  MManager Base Type                              -->
<!-- ################################################ -->
<complexType name="MManagerType">
    <sequence>
        <element name="managerCapabilityList"
type="mtdl:ManagerCapabilityListType" minOccurs="0"/>
        <element name="managementData" type="mtdl:ManagerDataBaseType"
minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="ManagerCapabilityListType">
    <sequence>
        <element name="managerCapability" type="mtdl:ManagerCapabilityType"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="listType" type="mtdl:ManagerCapabilityListEnumType"/>
</complexType>

<simpleType name="ManagerCapabilityListEnumType">
    <restriction base="string">
        <enumeration value="allManagerCapabilityList"/>
        <enumeration value="availableManagerCapabilityList"/>
        <enumeration value="appliedManagerCapabilityList"/>
    </restriction>
</simpleType>

<complexType name="ManagerCapabilityType">
    <sequence>
        <element name="managerCapabilityParameter"
type="mtdl:ManagerCapabilityParameterType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="managerCapabilityName" type="mpeg7:termReferenceType"
use="required"/>
</complexType>

<complexType name="ManagerCapabilityParameterType">
    <attribute name="managerCapabilityParameterName"
type="mpeg7:termReferenceType" use="required"/>
</complexType>

<complexType name="ManagerDataBaseType" abstract="true"/>
```


5.10.3 Binary Representation

MManagerType {	**Number of Bits**	**Mnemonic**
managerCapabilityListFlag	1	bslbf
managementDataFlag	1	bslbf
If(managerCapabilityListFlag) {		
ManagerCapabilityList		ManagerCapabilityListType
}		

MManagerType {	**Number of Bits**	**Mnemonic**
if(managementDataFlag) {		
managementData		ManagerDataBaseType
}		
}		

ManagerCapabilityListType {	**Number of Bits**	**Mnemonic**
listTypeFlag	1	bslbf
NumOfManagerCapability		vluimbsf5
if(k=0;k<NumOfManagerCapability;k++) {		
managerCapability[k]		ManagerCapabilityType
}		
if(listTypeFlag) {		
listType	2	bslbf
}		
}		

ManagerCapabilityType {	**Number of Bits**	**Mnemonic**
managerCapabilityParameterFlag	1	bslbf
if(managerCapabilityParameterFlag) {		
NumOfManagerCapabilityParameter		vluimsbf5
if(k=0;k<NumOfManagerCapabilityParameter;k++) {		
managerCapabilityParameter[k]		ManagerCapabilityParameterType
}		
}		
managerCapabilityName	8	bslbf
}		

5.10.4 Semantics

Name	*Definition*
`managerCapabilityListFlag`	This field, which is only present in the binary representation, indicates the presence of the `managerCapabilityList` element. If it is set to "1," the `managerCapabilityList` element follows.
`managementDataFlag`	This field, which is only present in the binary representation, indicates the presence of the `managementDataFlag` element. If it is set to "1," the `managementDataFlag` element follows.
`managerCapabilityList`	List of the MManager's capabilities.
`ManagerCapabilityListType`	Tool for describing manager capability.

Name	*Definition*
managementData	Management Data by an MManager. (Placeholder)
listTypeFlag	This field, which is only present in the binary representation, indicates the presence of the listType attribute. If it is set to "1," the listType attribute follows.
NumOfManagerCapability	This field, which is only present in the binary representation, specifies the number of managerCapability.
managerCapability	List of the MManager's full capabilities.
listType	Tool for specify the type of manager capabilities among allManagerCapabilityList, availableManagerCapabilityList, and appliedManagerCapabilityList. In the binary description, the mapping is used between the listType and the 2bit binary representation as follows: - allManagerCapabilityList: 00, - availableManagerCapabilityList: 01, - appliedManagerCapabilityList: 10, - Reserved: 11.
managerCapabilityListEnumType	It is a numeration list that can be one of the capability types.
ManagerCapabilityType	Tool for describing capabilities.
managerCapabilityParameterFlag	This field, which is only present in the binary representation, indicates the presence of the managerCapabilityParameter element. If it is set to "1," the managerCapabilityParameter element follows.
NumOfManagerCapabilityParameter	This field, which is only present in the binary representation, specifies the number of managerCapabilityParameter.
managerrCapabilityParameter	List of the MManager's capability parameters.
managerCapabilityName	Name of a capability of MManager. The type of capability shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the ManagerCapabilityCS defined in A.2.4.
ManagerCapabilityParameterType	Tool for describing capability parameters.
managerCapabilityParameterName	Name of a capability parameter. The type of the capability parameter shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the ManagerCapabilityParameterCS defined in A.3.5 (Table A.12).

Name	*Definition*
`ManagementDataBaseType`	Tool for describing management data. It is an abstract type of management data inherited by MManager. The abstract type of management data shall be described. (Placeholder)

5.10.5 Example

This example shows an `MManager` with a data instance ID of "MMN006-0001" and a device ID of "MMN006." This `MManager` has capabilities of "`MANAGER_RETRIVE_MTHINGS_BY_CAPABILITY`," "`MANAGER_REGISTER_MTHING`," and "`MANAGER_UNREGISTER_MTHING`."

```
<mtdl:MThingInfo id="MMN006-0001" idRef="MMN006">
    <mtdl:MManager>
        <mtdl:managerCapabilityList listType="allManagerCapabilityList">
            <mtdl:managerCapability managerCapabilityName="urn:mpeg:mpeg-IoMT:01-
ManagerCapabilityCS-NS:MANAGER_RETRIVE_MTHINGS_BY_CAPABILITY"/>
            <mtdl:managerCapability managerCapabilityName="urn:mpeg:mpeg-IoMT:01-
ManagerCapabilityCS-NS:MANAGER_REGISTER_MTHING"/>
            <mtdl:managerCapability managerCapabilityName="urn:mpeg:mpeg-IoMT:01-
ManagerCapabilityCS-NS:MANAGER_UNREGISTER_MTHING"/>
        </mtdl:managerCapabilityList>
    </mtdl:MManager>
</mtdl:MThingInfo>
```

5.11 Media aggregator description language

5.11.1 General

This subclause specifies tools for describing a media thing aggregator, which is composed of multiple individual MThings. The following subclause defines a complex type of `MAggregatorType`, which contains the MAggregator capability description along with its group of MThings inside.

5.11.2 Syntax

```
<!-- ########################################### -->
<!--  MAggregator Base Type                      -->
<!-- ########################################### -->
<complexType name="MAggregatorType">
    <sequence>
        <element name="aggregatorCapabilityList"
type="mtdl:AggregatorCapabilityListType" minOccurs="0"/>
        <element name="aggregatedMThingList" type="mtdl:AggregatedMThingListType"
minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="AggregatorCapabilityListType">
    <sequence>
        <element name="aggregatorCapability" type="mtdl:AggregatorCapabilityType"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="listType" type="mtdl:AggregatorCapabilityListEnumType"/>
```

```
</complexType>

<simpleType name="AggregatorCapabilityListEnumType">
    <restriction base="string">
        <enumeration value="allAggregatorCapabilityList"/>
        <enumeration value="availableAggregatorCapabilityList"/>
        <enumeration value="appliedAggregatorCapabilityList"/>
    </restriction>
</simpleType>

<complexType name="AggregatorCapabilityType">
    <sequence>
        <element name="aggregatorCapabilityParameter"
type="mtdl:AggregatorCapabilityParameterType" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
    <attribute name="aggregatorCapabilityName" type="mpeg7:termReferenceType"
use="required"/>
</complexType>

<complexType name="AggregatorCapabilityParameterType">
    <attribute name="aggregatorCapabilityParameterName"
type="mpeg7:termReferenceType" use="required"/>
</complexType>

<complexType name="AggregatedMThingListType">
    <sequence>
        <element name="MSensor" type="mtdl:MSensorType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="MActuator" type="mtdl:MActuatorType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="MAnalyser" type="mtdl:MAnalyserType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="MStorage" type="mtdl:MStorageType" minOccurs="0"
maxOccurs="unbounded"/>
        <element name="MManager" type="mtdl:MManagerType" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

5.11.3 Binary Representation

MAggregatorType {	**Number of Bits**	**Mnemonic**
aggregatorCapabilityListFlag	1	bslbf
aggregatedMThingListFlag	1	bslbf
If(aggregatorCapabilityListFlag) {		
aggregatorCapabilityList		AggregatorCapabilityListType
}		
if(aggregatedMThingListFlag) {		
aggregatedMThingList		AggregatedMThingListType
}		
}		

AggregatorCapabilityListType {	Number of Bits	Mnemonic
listTypeFlag	1	bslbf
NumOfAggregatorCapability		vluimbsf5
if(k=0;k<NumOfAggregatorCapability;k++) {		
aggregatorCapability[k]		AggregatorCapabilityType
}		
if(listTypeFlag) {		
listType	2	bslbf
}		
}		

AggregatorCapabilityType {	Number of Bits	Mnemonic
aggregatorCapabilityParameterFlag	1	bslbf
if(aggregatorCapabilityParameterFlag) {		
NumOfAggregatorCapabilityParameter		vluimsbf5
if(k=0;k<NumOfAggregatorCapabilityParameter;k++) {		
aggregatorCapabilityParameter[k]		AggregatorCapabilityParameterType
}		
}		
aggregatorCapabilityName	8	bslbf
}		

AggregatorCapabilityParameterType {	Number of Bits	Mnemonic
aggregatorCapabilityParameterName	8	bslbf
}		

AggregatedMThingListType {	Number of Bits	Mnemonic
MSensorFlag	1	bslbf
MActuatorFlag	1	bslbf
MAnalyserFlag	1	bslbf
MStorageFlag	1	bslbf
MManagerFlag	1	bslbf
if(MSensorFlag) {		
NumOfMSensor		vluimbsf5
if(k=0;k<NumOfMSensor;k++) {		
MSensor[k]		MSensorType
}		
}		
if(MActuatorFlag) {		
NumOfMActuator		vluimbsf5

AggregatedMThingListType {	Number of Bits	Mnemonic
if(k=0;k<NumOfMActuator;k++) {		
MActuator[k]		MActuatorType
}		
}		
if(MAnalyserFlag) {		
NumOfMAnalyser		vluimbsf5
if(k=0;k<NumOfMAnalyser;k++) {		
MAnalyser[k]		MAnalyserType
}		
}		
if(MStorageFlag) {		
NumOfMStorage		vluimbsf5
if(k=0;k<NumOfMStorage;k++) {		
MStorage[k]		MStorageType
}		
}		
if(MManagerFlag) {		
NumOfMManager		vluimbsf5
if(k=0;k<NumOfMManager;k++) {		
MManager[k]		MManagerType
}		
}		

5.11.4 Semantics

Name	Definition
`aggregatorCapabilityListFlag`	This field, which is only present in the binary representation, indicates the presence of the `aggregatorCapabilityList` element. If it is set to "1," the `aggregatorCapabilityList` element follows.
`aggregatedMThingListFlag`	This field, which is only present in the binary representation, indicates the presence of the `aggregatedMThingList` element. If it is set to "1," the `aggregatedMThingFlag` element follows.
`aggregatorCapabilityList`	List of the MAggregator's capabilities.
`aggregatedMThingList`	List of aggregated MThing list.
`AggregatorCapabilityListType`	Tool for describing aggregator capability.
`listTypeFlag`	This field, which is only present in the binary representation, indicates the presence of the `listType` attribute. If it is set to "1," the `listType` attribute follows.

Name	Definition
NumOfAggregatorCapability	This field, which is only present in the binary representation, specifies the number of aggregatorCapability.
aggregatorCapability	List of the MAggregator's full capabilities.
listType	Tool for specifying the type of aggregator capabilities among allAggregatorCapabilityList, availableAggregatorCapabilityList, and appliedAggregatorCapabilityList. In the binary description, the mapping is used between the listType and the 2bit binary representation as follows: - allAggregatorCapabilityList: 00, - availableAggregatorCapabilityList: 01, - appliedAggregatorCapabilityList: 10, - Reserved: 11.
aggregatorCapabilityListEnumType	It is a numeration list, which can be one of the capability types.
AggregatorCapabilityType	Tool for describing capabilities.
aggregatorCapabilityParameterFlag	This field, which is only present in the binary representation, indicates the presence of the aggregatorCapabilityParameter element. If it is set to "1," the aggregatorCapabilityParameter element follows.
NumOfAggregatorCapabilityParameter	This field, which is only present in the binary representation, specifies the number of aggregatorCapabilityParameter.
aggregatorCapabilityParameter	List of the MAggregator's capability parameters.
aggregatorCapabilityName	Name of a capability of MManager. The type of capability shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the AggregatorCapabilityCS defined in A.2.6.
AggregatorCapabilityParameterType	Tool for describing capability parameters.
aggregatorCapabilityParameterName	Name of a capability parameter. The type of the capability parameter shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the AggregatorCapabilityParameterCS defined in A.3.6 (Table A.13).

Name	Definition
AggregatedMThingListType	Tool for describing an aggregated MThing list.
MSensorFlag	This field, which is only present in the binary representation, indicates the presence of the MSensor element. If it is set to "1," the MSensor element follows.
MActuatorFlag	This field, which is only present in the binary representation, indicates the presence of the MActuator element. If it is set to "1," the MActuator element follows.
MAnalyserFlag	This field, which is only present in the binary representation, indicates the presence of the MAnalyser element. If it is set to "1," the MAnalyser element follows.
MStorageFlag	This field, which is only present in the binary representation, indicates the presence of the MStorage element. If it is set to "1," the MStorage element follows.
MManagerFlag	This field, which is only present in the binary representation, indicates the presence of the MManager element. If it is set to "1," the MManager element follows.
NumOfMSensor	This field, which is only present in the binary representation, specifies the number of MSensor.
NumOfMActuator	This field, which is only present in the binary representation, specifies the number of MActuator.
NumOfMAnalyser	This field, which is only present in the binary representation, specifies the number of MAnalyser.
NumOfMStorage	This field, which is only present in the binary representation, specifies the number of MStorage.
NumOfMManager	This field, which is only present in the binary representation, specifies the number of MManager.

5.11.5 Example

This example shows a MAggregator with a data instance ID of "MAG007-0001" and a device ID of "MAG007." This MAggregator has the capability of "AGGREGATOR_SHOW_AGGREGATED_MTHINGS," representing that the MAggregator is capable of showing the list of its MThing composition. This MAggregator has an MCamera and an MCameraActuator in it. The MCamera has the capability of "SENSOR_CAPTURE_VIDEO," representing that the corresponding MCamera can capture a video. The MCameraActuator has the capability of "ACTUATOR_CHANGE_RESOLUTION," representing that the corresponding MCameraActuator can change the camera resolution.

```
<mtdl:MThingInfo id="MAG007-0001" idRef="MAG007">
   <mtdl:MAggregator>
      <mtdl:aggregatorCapabilityList>
         <mtdl:aggregatorCapability aggregatorCapabilityName="urn:mpeg:mpeg-
IoMT:01-SensorCapabilityCS-NS:AGGREGATOR_SHOW_AGGREGATED_MTHINGS"/>
      <mtdl:aggregatorCapabilityList>
      <mtdl:aggregatedMThingList>
         <mtdl:MSensor>
            <mtdl:sensorCapabilityList>
               <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-
IoMT:01-SensorCapabilityCS-NS:SENSOR_CAPTURE_VIDEO"/>
            </mtdl:sensorCapabilityList>
         </mtdl:MSensor>
         <mtdl:MActuator>
            <mtdl:actuatorCapabilityList>
               <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-
IoMT:01-ActuatorCapabilityCS-NS:ACTUATOR_CHANGE_RESOLUTION"/>
            </mtdl:actuatorCapabilityList>
         </mtdl:MActuator>
      </mtdl:aggregatedMThingList>
   </mtdl:MAggregator>
</mtdl:MThingInfo>
```

Another example below shows the capabilities of smart glasses that are a typical example of a MAggregator. It is assumed that the smart glasses in the example contain several sensors, actuators, analysers, and storage. As shown in the example, the available capabilities of MSensors contained in the smart glasses are `SENSOR_CAPTURE_VIDEO`, `SENSOR_CAPTURE_AUDIO`, `SENSOR_STREAM_VIDEO`, `SENSOR_CAPTURE_ STEREO_VIDEO`, and `SENSOR_STREAM_STEREO_VIDEO`. Using these capability lists, other MThings in the network can discover the sensor functionalities of the smart glasses. After collecting the sensor functionalities of the smart glasses, a `sensorCapabilityParameter` depending on each capability is shown. The `sensorCapabilityParameter` presents the supportable video/audio compression methods and protocols such as `VIDEO_CODEC_MP4_AVC`, `AUDIO_CODEC_PCM`, and `VIDEO_STREAMING_PROTOCOL_HTTP`. For example, in this case, a sensor with the `SENSOR_CAPTURE_VIDEO` capability can encode a video using `VIDEO_CODEC_MP4_AVC`.

The MActuators of the smart glasses have capabilities of `ACTUATOR_DISPLAY_TRANSPARENT`, `ACTUATOR_VIBRATE`, `ACTUATOR_PLAY_AUDIO`, and `ACTUATOR_PLAY_VIDEO`.

Besides, the capabilities of MAnalysers contained in the smart glasses are `ANALYSER_DETECT_HAND_GESTURE`, `ANALYSER_RECOGNISE_HAND_GESTURE`, `ANALYSER_MAP_HAND_GESTURE_COMMAND`, `ANALYSER_RECOGNISE_SPEECH`, and `ANALYSER_ANALYSE_QUESTION`. The `allAnalyserCapabilityList` presents all capabilities of MAnalysers in the smart glasses supporting other MThings. And the storage of the smartglasses has the capability of "STORAGE_SAVE," which can store "MP4" files.

```
<mtdl:MThingInfo ID="SMGLASS-20210001" idRef="SMGLASS">
  <mtdl:MAggregator>
    <mtdl:aggregatorCapabilityList listType="availableAggregatorCapabilityList">
      <mtdl:aggregatorCapability aggregatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:AGGREGATOR_SHOW_AGGREGATED_MTHINGS"/>
    </mtdl:aggregatorCapabilityList>
    <mtdl:aggregatedMThingList>
      <!--MSensor-->
      <mtdl:MSensor>
        <mtdl:sensorCapabilityList listType="availableSensorCapabilityList">
          <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:SENSOR_CAPTURE_VIDEO">
```

```
            <mtdl:sensorCapabilityParameter
sensorCapabilityParameterName="urn:mpeg:mpeg-IoMT:01-SensorCapabilityParameterCS-
NS:VIDEO_CODEC_MP4_AVC"/>
          </mtdl:sensorCapability>
          <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:SENSOR_CAPTURE_AUDIO">
            <mtdl:sensorCapabilityParameter
sensorCapabilityParameterName="urn:mpeg:mpeg-IoMT:01-SensorCapabilityParameterCS-
NS:AUDIO_CODEC_PCM"/>
          </mtdl:sensorCapability>
          <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:SENSOR_STREAM_VIDEO">
            <mtdl:sensorCapabilityParameter
sensorCapabilityParameterName="urn:mpeg:mpeg-IoMT:01-SensorCapabilityParameterCS-
NS:VIDEO_STREAMING_PROTOCOL_HTTP"/>
          </mtdl:sensorCapability>
          <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:SENSOR_CAPTURE_STEREO_VIDEO">
            <mtdl:sensorCapabilityParameter
sensorCapabilityParameterName="urn:mpeg:mpeg-IoMT:01-SensorCapabilityParameterCS-
NS:VIDEO_CODEC_MP4_AVC"/>
          </mtdl:sensorCapability>
          <mtdl:sensorCapability sensorCapabilityName="urn:mpeg:mpeg-IoMT:01-
SensorCapabilityCS-NS:SENSOR_STREAM_STEREO_VIDEO">
            <mtdl:sensorCapabilityParameter
sensorCapabilityParameterName="urn:mpeg:mpeg-IoMT:01-SensorCapabilityParameterCS-
NS:VIDEO_STREAMING_PROTOCOL_HTTP"/>
          </mtdl:sensorCapability>
        </mtdl:sensorCapabilityList>
      </mtdl:MSensor>
      <!--/MSensor-->
      <!--MActuator-->
      <mtdl:MActuator>
        <mtdl:actuatorCapabilityList listType="allActuatorCapabilityList">
          <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
ActuatorCapabilityCS-NS:ACTUATOR_DISPLAY_TRANSPARENT"/>
          <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
ActuatorCapabilityCS-NS:ACTUATOR_VIBRATE"/>
          <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
ActuatorCapabilityCS-NS:ACTUATOR_PLAY_AUDIO"/>
          <mtdl:actuatorCapability actuatorCapabilityName="urn:mpeg:mpeg-IoMT:01-
ActuatorCapabilityCS-NS:ACTUATOR_PLAY_VIDEO"/>
        </mtdl:actuatorCapabilityList>
      </mtdl:MActuator>
      <!--/MActuator-->
      <!--MAnalyser-->
      <mtdl:MAnalyser>
        <mtdl:analyserCapabilityList listType="allAnalyserCapabilityList">
          <mtdl:analyserCapability analyserCapabilityName="urn:mpeg:mpeg-IoMT:01-
AnalyserCapabilityCS-NS:ANALYSER_DETECT_HAND_GESTURE"/>
          <mtdl:analyserCapability analyserCapabilityName="urn:mpeg:mpeg-IoMT:01-
AnalyserCapabilityCS-NS:ANALYSER_RECOGNISE_HAND_GESTURE"/>
          <mtdl:analyserCapability analyserCapabilityName="urn:mpeg:mpeg-IoMT:01-
AnalyserCapabilityCS-NS:ANALYSER_MAP_HAND_GESTURE_COMMAND"/>
          <mtdl:analyserCapability analyserCapabilityName="urn:mpeg:mpeg-IoMT:01-
AnalyserCapabilityCS-NS:ANALYSER_RECOGNISE_SPEECH"/>
          <mtdl:analyserCapability analyserCapabilityName="urn:mpeg:mpeg-IoMT:01-
AnalyserCapabilityCS-NS:ANALYSER_ANALYSE_QUESTION"/>
        </mtdl:analyserCapabilityList>
      </mtdl:MAnalyser>
      <!--/MAnalyser-->
```

```
      <!--MStorage-->
      <mtdl:MStorage>
        <mtdl:storageCapabilityList listType="availableStorageCapabilityList">
          <mtdl:storageCapability storageCapabilityName="urn:mpeg:mpeg-IoMT:01-
StorageCapabilityCS-NS:STORAGE_SAVE">
            <mtdl:storageCapabilityParameter
storageCapabilityParameterName="urn:mpeg:mpeg-IoMT:01-
StorageCapabilityParameterCS-NS:MP4"/>
          </mtdl:storageCapability>
        </mtdl:storageCapabilityList>
      </mtdl:MStorage>
      <!--/MStorage-->
    </mtdl:aggregatedMThingList>
  </mtdl:MAggregator>
</mtdl:MThingInfo>
```

6 Media sensor output vocabulary

6.1 General

This subclause specifies syntax and semantics of the media sensor output vocabulary, which comprises the following media sensors:

— IoMT sensed data captured time.

NOTE MSOV has been designed in an extensible way, and additional media sensors can be added easily.

EXAMPLE Additional media sensors can be added as extensions to `mtdl:SensedDataBaseType` and conformance to MTDL.

6.2 Schema wrapper

The syntax of description tools specified in this subclause is provided as a collection of schema components, including type definitions and element declarations. To form a valid schema document, users can gather these schema components in the same document with the following declaration defining, in particular, the target namespace and the namespaces prefixes.

```
<?xml version="1.0"?>
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mtdl="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
  xmlns:msov="cmpeg:mpeg-IoMT:2021:01-MSOV-NS"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
  targetNamespace="urn:mpeg:mpeg-IoMT:2021:01-MSOV-NS"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="ISO/IEC 23093-3" id="MPEG-IOMT-MSOV.xsd">

  <import namespace="urn:mpeg:mpeg7:schema:2004"
schemaLocation="http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
7_schema_files/mpeg7-v2.xsd"/>
  <import namespace="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
schemalocation=" http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
IOMT_schema_files/MPEG-IOMT-MTDL.xsd"/>
```

The following line should be appended to the resulting schema document to obtain a well-formed XML document.

```
</schema>
```

6.3 IoMT sensed data captured time

6.3.1 General

This subclause specifies data formats to describe the sensed data captured time that IoMT sensors can produce. The `CapturedTime` is a Gregorian date-time that capturing of sensed data.

6.3.2 Syntax

```
<!-- ################################################ -->
<!-- Data formats for IoMT sensed data caputured time -->
<!-- ################################################ -->
<complexType name="CapturedTimeType">
   <complexContent>
      <extension base="mtdl:SensedDataBaseType">
         <sequence>
            <element name="CapturedTime" type="mpeg7:TimePointType" />
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

6.3.3 Binary Representation

CapturedTimeType {	**Number of Bits**	**Mnemonic**
SensedDataBaseType		SensedDataBaseType
CapturedTime		mpeg7:TimePointType
}		

6.3.4 Semantics

Name	*Definition*
`CapturedTimeType`	Tool for specifying the time point at which capturing sensed data started based on Gregorian date time and time zone referring to ISO 8601.
`CapturedTime`	It describes the captured time of sensed data.

6.3.5 Example

This example shows how the captured time of sensed data from an MSesnor can be described.

```
<mtdl:MThingInfo>
    <mtdl:MSensor>
        <mtdl:sensedData xsi:type="msov:CapturedTimeType">
            <msov:CapturedTime>2021-09-07T10:47+09:00</msov:CapturedTime>
        </mtdl:sensedData>
    </mtdl:MSensor>
</mtdl:MThingInfo>
```

7 Media actuator command vocabulary

7.1 General

This subclause specifies syntax and semantics of the media actuator command vocabulary, which comprises the following media actuators:

— IoMT speaker;

— IoMT display;

— IoMT camera actuator;

— IoMT camera light.

NOTE MACV has been designed in an extensible way, and additional media actuators can be added easily.

EXAMPLE Additional media actuators can be added as extensions to `mtdl:ActuationDataBaseType` and conformance to MTDL.

7.2 Schema wrapper

The syntax of description tools specified in this subclause is provided as a collection of schema components, including type definitions and element declarations. To form a valid schema document, users can gather these schema components in the same document with the following declaration defining, in particular, the target namespace and the namespaces prefixes.

```
<?xml version="1.0"?>
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mtdl="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
  xmlns:macv="urn:mpeg:mpeg-IoMT:2021:01-MACV-NS"
  xmlns:scdv="urn:mpeg:mpeg-v:2017:01-SCDV-NS"
  targetNamespace="urn:mpeg:mpeg-IoMT:2021:01-MACV-NS"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="ISO/IEC 23093-3" id="MPEG-IOMT-MACV.xsd">

   <import namespace="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
schemaLocation="http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
IOMT_schema_files/MPEG-IOMT-MTDL.xsd"/>
  <import namespace="urn:mpeg:mpeg-v:2017:01-SCDV-NS"
schemalocation="http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
V_schema_files/MPEG-V-SCDV.xsd"/>
```

The following line should be appended to the resulting schema document to obtain a well-formed XML document.

```
</schema>
```

7.3 IoMT speaker

7.3.1 General

This subclause specifies data formats to describe the commands that can control the IoMT speaker. `MSpeaker` obtains audio sources from `MMicrophone`, `MStorage`, etc. According to options, `MSpeaker` plays an audio source and adjusts volumes.

7.3.2 Syntax

```
<!-- ################################################ -->
<!-- Data formats for IoMT speaker commands           -->
<!-- ################################################ -->
<complexType name="AudioPlayType">
    <complexContent>
        <extension base="mtdl:ActuationDataBaseType">
            <sequence>
                <element name="playType" type="macv:AudioActuationType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<simpleType name="AudioActuationType">
    <restriction base="string">
        <enumeration value="play"/>
        <enumeration value="stop"/>
        <enumeration value="pause"/>
        <enumeration value="rewind2x"/>
        <enumeration value="rewind4x"/>
        <enumeration value="rewind8x"/>
        <enumeration value="rewind16x"/>
        <enumeration value="rewind32x"/>
        <enumeration value="fastForward2x"/>
        <enumeration value="fastForward4x"/>
        <enumeration value="fastForward8x"/>
        <enumeration value="fastForward16x"/>
        <enumeration value="fastForward32x"/>
    </restriction>
</simpleType>

<complexType name="setVolumeType">
    <complexContent>
        <extension base="mtdl:ActuationDataBaseType">
            <sequence>
                <element name="volume">
                    <simpleType>
                        <restriction base="integer">
                            <minInclusive value="0"/>
                            <maxInclusive value="100"/>
```

```
                    </restriction>
                </simpleType>
            </element>
        </sequence>
    </extension>
</complexContent>
</complexType>
```

7.3.3 Binary Representation

AudioPlayType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
playType	4	bslbf
}		

setVolumeType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
volume	7	uimbsf
}		

7.3.4 Semantics

Name	Definition
AudioPlayType	Tool for describing a command for a speaker device to follow.
playType	It describes the audio play command. In the binary description, the mapping is used between the audio playType and the 4bit binary representation as follows: - stop: 0000 - play: 0001 - pause: 0010 - rewind2x: 0011 - rewind4x: 0100 - rewind8x: 0101 - rewind16x: 0110 - rewind32x: 0111 - fastForward2x: 1000 - fastForward4x: 1001 - fastForward8x: 1010 - fastForward16x: 1011 - fastForward32x: 1100 - reserved: 1101 - 1111
AudioActuationType	It describes the commands: play, stop, rewind 2x to 32x, and fast forward 2x to 32x.
setVolumeType	Tool for setting the volume of the speaker.
volume	It describes the volume to a speaker in a percentage between 0 and 100.

7.3.5 Example

This example shows how to set fast forward at 2x speed of an IoMT speaker.

```
<mtdl:actuationData xsi:type="macv:AudioPlayType">
     <macv:playType>fastForward2x</macv:playType>
</mtdl:actuationData>
```

This example shows how to set the volume 80 % of an IoMT speaker.

```
<mtdl:actuationData xsi:type="macv:setVolumeType">
     <macv:volume>80</macv:volume>
</mtdl:actuationData>
```

7.4 IoMT display

7.4.1 General

This subclause specifies data formats to describe the commands that control the IoMT display. `MDisplay` obtains video sources from `MCamera`, `MStorage`, etc. According to options, `MDisplay` plays a video source and changes brightness.

7.4.2 Syntax

```
<!-- ############################################## -->
<!-- Data formats for IoMT display commands         -->
<!-- ############################################## -->
<complexType name="VideoPlayType">
   <complexContent>
      <extension base="mtdl:ActuationDataBaseType">
         <sequence>
            <element name="playType" type="macv:VideoActuationType"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<simpleType name="VideoActuationType">
   <restriction base="string">
      <enumeration value="stop"/>
      <enumeration value="play"/>
      <enumeration value="pause"/>
      <enumeration value="rewind2x"/>
      <enumeration value="rewind4x"/>
      <enumeration value="rewind8x"/>
      <enumeration value="rewind16x"/>
      <enumeration value="rewind32x"/>
      <enumeration value="fastForward2x"/>
      <enumeration value="fastForward4x"/>
      <enumeration value="fastForward8x"/>
      <enumeration value="fastForward16x"/>
      <enumeration value="fastForward32x"/>
   </restriction>
</simpleType>
```

```
<complexType name="setBrightnessType">
    <complexContent>
        <extension base="mtdl:ActuationDataBaseType">
            <sequence>
                <element name="brightness">
                    <simpleType>
                        <restriction base="integer">
                            <minInclusive value="0"/>
                            <maxInclusive value="100"/>
                        </restriction>
                    </simpleType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

7.4.3 Binary Representation

VideoPlayType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
playType	4	bslbf
}		

setBrightnessType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
brightness	7	uimbsf
}		

7.4.4 Semantics

Name	Definition
VideoPlayType	Tool for describing a command for a display device to follow.
playType	It describes the video play command. In the binary description, the mapping is used between the video playType and the 4bit binary representation as follows: - stop: 0000 - play: 0001 - pause: 0010 - rewind2x: 0011 - rewind4x: 0100 - rewind8x: 0101 - rewind16x: 0110 - rewind32x: 0111 - fastForward2x: 1000 - fastForward4x: 1001 - fastForward8x: 1010 - fastForward16x: 1011 - fastForward32x: 1100 - reserved: 1101 - 1111
VideoActuationType	It describes the commands: play, stop, rewind 2x to 32x, and fast forward 2x to 32x.
setBrightnessType	Tool for setting the brightness of the display.
brightness	It describes the brightness of the display.

7.4.5 Example

This example shows how to set “stop playing” of an IoMT display.

```
<mtdl:actuationData xsi:type="macv:VideoPlayType">
    <macv:playType>stop</macv:playType>
</mtdl:actuationData>
```

This example shows how to set brightness to 100 % of an IoMT display.

```
<mtdl:actuationData xsi:type="macv:setBrightnessType">
    <macv:brightness>100</macv:brightness>
</mtdl:actuationData>
```

7.5 IoMT camera actuator

7.5.1 General

This subclause specifies data formats to describe the commands that can control the IoMT camera actuator. `MCameraActuator` obtains options to set or change the camera state. According to options, the `MCameraActuator` sets the camera’s orientation, zoom or resolution.

7.5.2 Syntax

```
<!-- ############################################### -->
<!-- Definition of IoMT camera actuation commands    -->
<!-- ############################################### -->
<complexType name="setCameraOrientationType">
    <complexContent>
        <extension base="mtdl:ActuationDataBaseType">
            <sequence>
                <element name="yaw">
                    <simpleType>
                        <restriction base="integer">
                            <minInclusive value="-180"/>
                            <maxInclusive value="180"/>
                        </restriction>
                    </simpleType>
                </element>
                <element name="pitch">
                    <simpleType>
                        <restriction base="integer">
                            <minInclusive value="-180"/>
                            <maxInclusive value="180"/>
                        </restriction>
                    </simpleType>
                </element>
                <element name="roll">
                    <simpleType>
                        <restriction base="integer">
                            <minInclusive value="-180"/>
                            <maxInclusive value="180"/>
                        </restriction>
                    </simpleType>
                </element>
```

```
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="setCameraZoomType">
    <complexContent>
        <extension base="mtdl:ActuationDataBaseType">
            <sequence>
                <element name="zoom" type="float"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="setCameraResolutionType">
    <complexContent>
        <extension base="mtdl:ActuationDataBaseType">
            <sequence>
                <element name="resolution" type="scdv:ResolutionType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

7.5.3 Binary Representation

setCameraOrientationType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
yaw	9	simsbf
pitch	9	simsbf
roll	9	simsbf
}		

setCameraZoomType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
zoom	32	fsfb
}		

setCameraResolutionType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
resolution		ResolutionType ISO/IEC 23005-5:2016(E)
}		

7.5.4 Semantics

Name	Definition
setCameraOrientationType	Tool for describing a command to set a camera orientation.
yaw	It describes the angle to rotate in the z-axis, Ψ(yaw) in degrees between -180 and 180.
pitch	It describes the angle to rotate in the y-axis, θ(pitch) in degrees between -180 and 180.
roll	It describes the angle to rotate in the x-axis, φ (roll), in degrees between -180 and 180.
setCameraZoomType	Tool for a command to set the camera zoom.
zoom	It describes the magnitude of zooming. The number less than 1 indicates the zoom-out, and the number greater than 1 indicates the zoom-in.
setCameraResolutionType	Tool for setting the camera resolution.
resolution	It describes the capturing resolution, which is composed of a width and a height.

7.5.5 Example

This example shows how to set the orientation of an IoMT camera to its the yaw angle 90°, the pitch angle 45°, and the roll angle 60°.

```
<mtdl:actuationData xsi:type="macv:setCameraOrientationType">
    <macv:yaw>90</macv:yaw>
    <macv:pitch>45</macv:pitch>
    <macv:roll>60</macv:roll>
</mtdl:actuationData>
```

This example shows how to set the zoom-in to 4 times.

```
<mtdl:actuationData xsi:type="macv:setCameraZoomType">
    <macv:zoom>4</macv:zoom>
</mtdl:actuationData>
```

This example shows how to set the camera resolution to 1920x1080.

```
<mtdl:actuationData xsi:type="macv:setCameraResolutionType">
    <macv:resolution>
        <scdv:Width>1920</scdv:Width>
        <scdv:Height>1080</scdv:Height>
    </macv:resolution>
</mtdl:actuationData>
```

7.6 IoMT light

7.6.1 General

This subclause specifies data formats to describe the commands that can control the IoMT light. The `MLight` obtains options to set or change the light state. According to options, `MLight` sets light's brightness, saturation, or hue.

7.6.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of IoMT Light actuation commands      -->
<!-- ################################################ -->
<complexType name="setColorLightType">
   <complexContent>
      <extension base="mtdl:ActuationDataBaseType">
         <sequence>
            <element name="brightness" minOccurs="0">
               <simpleType>
                  <restriction base="float">
                     <minInclusive value="0.0"/>
                     <maxInclusive value="1.0"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="saturation" minOccurs="0">
               <simpleType>
                  <restriction base="float">
                     <minInclusive value="0.0"/>
                     <maxInclusive value="1.0"/>
                  </restriction>
               </simpleType>
            </element>
            <element name="hue" minOccurs="0">
               <simpleType>
                  <restriction base="integer">
                     <minInclusive value="0"/>
                     <maxInclusive value="360"/>
                  </restriction>
               </simpleType>
            </element>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```


7.6.3 Binary Representation

setColorLightType {	**Number of Bits**	**Mnemonic**
ActuationDataBaseType		ActuationDataBaseType
brightnessFlag	1	bslbf
saturationFlag	1	bslbf
hueFlag	1	bslbf

setColorLightType {	Number of Bits	Mnemonic
if(brightnessFlag) {		
brightness	32	fsfb
}		
if(saturationFlag) {		
saturation	32	fsfb
}		
if(hueFlag) {		
hue	9	uimsbf
}		
}		

7.6.4 Semantics

Name	Definition
brightnessFlag	This field, which is only present in the binary representation, indicates the presence of the brightness element. If it is set to "1," the brightness element follows.
saturationFlag	This field, which is only present in the binary representation, indicates the presence of the saturation element. If it is set to "1," the saturation element follows.
hueFlag	This field, which is only present in the binary representation, indicates the presence of the hue element. If it is set to "1," the hue element follows.
setColorLightType	Tool for describing a command to set the colour of lights.
brightness	It describes the value of the brightness in the HSV colour model.
saturation	It describes the value of the saturation in the HSV colour model.
hue	It describes the value of the hue in the HSV colour model.

7.6.5 Example

This example shows a command to set color of a light with the brightness 0.7, the saturation 0.8, and the hue 240.

```
<mtdl:actuationData xsi:type="macv:setColorLightType">
    <macv:brightness>0.7</macv:brightness>
    <macv:saturation>0.8</macv:saturation>
    <macv:hue>240</macv:hue>
</mtdl:actuationData>
```

This example shows a command to set color of a light with only the brightness 0.8.

```
<mtdl:actuationData xsi:type="macv:setColorLightType">
    <macv:brightness>0.8</macv:brightness>
</mtdl:actuationData>
```

This example shows a command to set color of a light with the saturation 0.9 and the hue 270.

```
<mtdl:actuationData xsi:type="macv:setColorLightType">
    <macv:saturation>0.9</macv:saturation>
    <macv:hue>270</macv:hue>
</mtdl:actuationData>
```


8 Media analyser output vocabulary

8.1 General

This subclause specifies syntax and semantics of the media analyser output vocabulary, which comprises the following media analysers:

— IoMT time synchronizer;

— IoMT social event detector;

— IoMT hand gesture detector;

— IoMT hand gesture recogniser;

— IoMT hand gesture command generator;

— IoMT healthcare information generator;

— IoMT odour image scent recogniser;

— IoMT question analyser;

— IoMT music frequency analyser;

— IoMT video content class generator;

— IoMT face region detector;

— IoMT face verifier;

— IoMT security title generator;

— IoMT light colour converter.

NOTE MAOV has been designed in an extensible way, and additional media analysers can be added easily.

EXAMPLE Additional media analysers can be added as extensions to `mtdl:AnalysedDataBaseType` and conformance to MTDL.

8.2 Schema wrapper

The syntax of description tools specified in this subclause is provided as a collection of schema components, including type definitions and element declarations. To form a valid schema document, users can gather these schema components in the same document with the following declaration defining, in particular, the target namespace and the namespaces prefixes.

```
<?xml version="1.0"?>
  <schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:mtdl="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
  xmlns:maov="urn:mpeg:mpeg-IoMT:2021:01-MAOV-NS"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
  targetNamespace="urn:mpeg:mpeg-IoMT:2021:01-MAOV-NS"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="ISO/IEC 23093-3" id="MPEG-IOMT-MAOV.xsd">

   <import namespace="urn:mpeg:mpeg-IoMT:2021:01-MTDL-NS"
schemaLocation="http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
IOMT_schema_files/MPEG-IOMT-MTDL.xsd"/>
  <import namespace="urn:mpeg:mpeg7:schema:2004"
schemaLocation="http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
7_schema_files/mpeg7-v2.xsd"/>
```

The following line should be appended to the resulting schema document to obtain a well-formed XML document.

```
</schema>
```

8.3 IoMT time synchroniser

8.3.1 General

This subclause specifies data formats to describe the outputs that the `MTimeSynchroniser` can produce. The `MTimeSynchroniser` obtains video sources or audio sources from two different `MCameras` with `MMicrophone` and calculates the time offset between two other videos. The `SyncedVideoType` is used to represent information for synchronising two different videos.

8.3.2 Syntax

```
<!-- ################################-->
<!--  Definition of Synced Video Type    -->
<!-- ################################-->

<complexType name="SyncedVideoType">
   <complexContent>
      <extension base="mtdl:AnalysedDataBaseType">
         <sequence>
            <choice>
               <sequence>
                  <element name="ReferenceVideoURL" type ="anyURI"/>
                  <element name="TargetVideoURL" type="anyURI"/>
               </sequence>
               <sequence>
```

```
                    <element name="ReferenceAudioURL" type = "anyURI" />
                    <element name="TargetAudioURL" type="anyURI"/>
                </sequence>
            </choice>
            <element name="TimeOffset" type="mpeg7:MediaTimeType"/>
        </sequence>
    </extension>
  </complexContent>
</complexType>
```

8.3.3 Binary Representation

SynchedVideoType {	**Number of bits**	**Mnemonic**
AnalysedDataBaseType		AnalysedDataBaseType
VideoAudioChoiceFlag	*1*	bslbf
If (VideoAudioChoice) {		
ReferenceVideoURL		UTF-8
TargetVideoURL		UTF-8
}		
Else {		
ReferenceAudioURL		UTF-8
TargetAudioURL		UTF-8
}		
TimeOffset	*32*	mpeg7:MediaTimeType
}		

8.3.4 Semantics

Name	**Definition**
`VideoAudioChoiceFlag`	This field, which is only present in the binary representation, indicates the choice of `ReferenceVideoURL` and `TargetVieoURL` or `ReferenceAudioURL` and `TargetAudioURL` elements. If it is set to "1," `ReferenceVideo` and `TargetVideoURL` elements follow. Otherwise, `ReferenceAudioURL` and `TargetAudioURL` elements follow.
`SyncedVideoType`	Tool for describing information of video with time information for synchronising two different videos.
`ReferenceVideoURL`	It describes a URL that identifies the reference video source.
`TargetVideoURL`	It describes a URL that identifies the target video source.
`ReferenceAudioURL`	It describes a URL that identifies the reference audio source.
`TargetAudioURL`	It describes a URL that identifies the target audio source.
`TimeOffset`	It describes the time offset of media for time synchronization using video, audio, and media time information defined in ISO/IEC 15938-5.

8.3.5 Example

This example shows the time offset between *con1.avi* and *con2.avi*. The *con2.avi* is 15 frames faster than the *con1.avi*. The time offset represents a difference of 15 frames on a 1/30 frame basis.

```
<mtdl:analysedData xsi:type="maov:SyncedVideoType">
    <maov:ReferenceVideoURL>http://www.moeg.com/con1.avi</maov:ReferenceVideoURL>
    <maov:TargetVideoURL>http://www.moeg.com/con2.avi</maov:TargetVideoURL>
    <maov:TimeOffset>
        <mpeg7:MediaRelIncrTimePoint
mediaTimeUnit="PT1N30F">15</mpeg7:MediaRelIncrTimePoint>
    </maov:TimeOffset>
</mtdl:analysedData>
```

8.4 IoMT social event detector

8.4.1 General

This subclause specifies the data format for a social event analyser output. The `MSocialEventDetector` obtains a video source from `MCamera` and analyses the social event of the video source. The `SocialEventType` is used to represent information about the analysed social event.

8.4.2 Syntax

```
<!-- ##########################################-->
<!-- Definition of Social Event Detector Type    -->
<!-- ##########################################-->

<complexType name="SocialEventType">
    <complexContent>
        <extension base="mtdl:AnalysedDataBaseType">
            <sequence>
                <element name="SocialEventName" type="mpeg7:termReferenceType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

8.4.3 Binary Representation

SocialEventType {	**Number of bits**	**Mnemonic**
AnalysedDataBaseType		AnalysedDataBaseType
SocialEventName	8	bsblf
}		

8.4.4 Semantics

Name	Definition
SocialEventType	Tool for describing the analysed social event.
SocialEventName	It describes the analysed social event type of a video source. The name of the social event shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the SocialEventCS defined in A.4.2 (Table A.15).

8.4.5 Example

This example shows that the detected social event is the birthday.

```
<mtdl:analysedData xsi:type="maov:SocialEventType">
    <maov:SocialEventName>urn:mpeg:mpeg-IoMT:01-SocialEventCS-
NS:BIRTHDAY_SOCIAL_EVENT</maov:SocialEventName>
</mtdl:analysedData>
```

8.5 IoMT hand gesture detector

8.5.1 General

This subclause specifies the data format for a hand gesture detector output. The MHandGestureDetector obtains video sources, including hand gestures from MCameras The HandGestureType represents detected hand gesture contours.

8.5.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of Hand Gesture Type                  -->
<!-- ################################################ -->
<complexType name="HandGestureType">
    <complexContent>
        <extension base="mtdl:AnalysedDataBaseType">
            <choice>
                <element name="HandContour" type="maov:HandContourType"/>
                <element name="HandTrajectory" type="maov:HandTrajectoryType"/>
            </choice>
            <attribute name="lefthand" type="boolean" default="false"/>

            <attribute name="frameRate" type="decimal" use="optional"
default="30"/>
        </extension>
    </complexContent>
</complexType>

<complexType name="HandContourType">
    <sequence>
        <choice>
            <element name="coordinate" type="mpeg7:integerVector" minOccurs="0"
maxOccurs="unbounded"/>
```

```
            <element name="GroupBezierCurve" type="maov:GroupBezierCurveType"
minOccurs="0" maxOccurs="unbounded"/>
        </choice>
        <element name="CenterPosition" type="mpeg7:integerVector" minOccurs="0"/>
    </sequence>
    <attribute name="useLastContour" type="boolean" default="false"/>
</complexType>

<complexType name="GroupBezierCurveType">
    <sequence >
        <element name="InitialStartPoint" type="mpeg7:integerVector"/>
        <sequence >
            <element name="BezierCurve" type="maov:BezierCurveType"
maxOccurs="unbounded"/>
        </sequence>
    </sequence>
    <attribute name="FittingError" type="double" />
</complexType>

<complexType name="BezierCurveType">
    <sequence>
        <element name="StartEndPoint" type="mpeg7:integerVector"/>
        <sequence>
            <element name="ControlPoint" type="mpeg7:integerVector"
maxOccurs="unbounded"/>
        </sequence>
    </sequence>
<attribute name="OrderOfBezierCurve" type="mpeg7:unsigned8" default="3"/>
</complexType>

<complexType name="HandTrajectoryType">
    <choice>
        <element name="GroupBezierCurve" type="maov:GroupBezierCurveType"
minOccurs="0"/>
        <sequence minOccurs="0" maxOccurs="unbounded">
            <element name="CenterPosition" type="mpeg7:integerVector"
minOccurs="0"/>
        </sequence>
    </choice>
    <attribute name="trajectoryType" type="boolean" default="false"/>
</complexType>
```


8.5.3 Binary Representation

HandGestureType {	Number of bits	Mnemonic
AnalysedDataBaseType		AnalysedDataBaseType
frameRateFlag	1	bslbf
HandGestureChoiceFlag	1	bslbf
If (HandGestureChoiceFlag) {		
HandContour		HandContourType
}		
else {		
HandTrajectory		HandTrajectoryType
}		

HandGestureType {	Number of bits	Mnemonic
lefthand	1	bslbf
If (frameRateFlag) {		
frameRate	8	uimsbf
}		
}		

HandContourType {	Number of bits	Mnemonic
useLastContourFlag	1	bslbf
handContourChoiceFlag	1	bslbf
GroupBezierCurveFlag	1	bslbf
centerPositionFlag	1	bslbf
If (handCountourChoiceFlag) {		
For (i=0; i< NbrOfHandGestureFrame; i++) {		
coordinate[i]	16*2	uimsbf
}		
}		
Else {		
For (i=0; i< NbrOfHandGestureFrame; i++) {		
GroupBezierCurve[i]		GroupBezierCurveType
}		
}		
}		
If (centerPositionFlag) {		
For (i=0; i< NbrOfHandGestureFrame; i++) {		
CenterPosition[i]	16*2	uimsbf
}		
}		
If (useLastContourFlag) {		
useLastContour	1	bslbf
}		
}		

GroupBezierCurveType {	Number of bits	Mnemonic
FittingErrorFlag	1	bslbf
InitialStartPoint	16 * 2	uimsbf
for (j=0; j<NbrOfHandGestureFrame; j++) {		
BezierCurve[j]		BezierCurveType
}		
If (FittingErrorFlag) {		
FittingError	32	fsfb
}		
}		

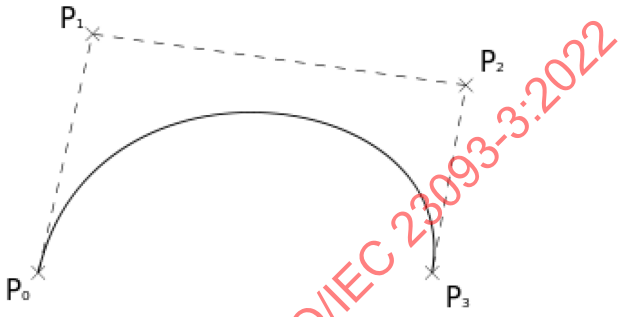
BezierCurveType{	Number of bits	Mnemonic
OrderOfBezierCurve	8	uimsbf
StartEndPoint	16 * 2	uimsbf
for(j=0; j< OrderOfBezierCurve; j++) {		
ControlPoint[j]	16 * 2	uimsbf
}		
}		

HandTrajectoryType {	Number of bits	Mnemonic
trajectoryTypeFlag	1	bslbf
HandTrajectoryChoiceFlag	1	bslbf
If (HandTrajectoryChoiceFlag) {		
for (i=0; i< NbrOfHandGestureFrame; i++) {		
CenterPosition[i]	16* 2	uimsbf
}		
}		
Else {		
GroupBezierCurve		GroupBezierCurveType
}		
If (trajectoryTypeFlag){		
trajectoryType	1	bslbf
}		
}		

8.5.4 Semantics

Name	Definition
`HandGestureType`	Tool for describing a hand gesture. The hand gesture description is used for the representation of a hand contour or hand trajectory.
`HandContour`	It describes a hand contour extracted from the incoming image sequences by the processing of hand gesture detection. The hand contour description can be used to generate hand gesture-based commands for user interaction.
`HandTrajectory`	It describes a hand trajectory extracted from the incoming image sequences by the processing of hand trajectory extraction. The hand trajectory description can be used to generate hand gesture-based commands for user interaction.
`lefthand`	It describes the indication of whether the left hand is used for generating hand gestures. If this value is "false," then the right hand is used for generating a hand gesture command; otherwise, the left hand is used. The default is "false."
`frameRateFlag`	This field, which is only present in the binary representation, indicates the presence of the `frameRate` element. If it is set to "1," the `frameRate` element follows.

Name	Definition
frameRate	It describes the frame rate of the incoming image sequence. The framerate is used to calculate the time interval between consecutive hand contours or the speed of hand motion, etc.
HandGestureChoiceFlag	This field, which is only present in the binary representation, indicates the choice of the HandCountour element or the HandTrajectory element. If it is set to "1," the HandCountour element follows; otherwise, the HandTrajectory element follows.
HandContourType	Tool for describing a hand contour. A hand contour description can represent a general hand contour, from which gesture-based interaction commands can be generated due to gesture recognition.
coordinate	The origin of the coordinate is the top-leftmost corner. It describes a hand contour in the form of a set of coordinates in 2-D space. The first value indicates the position of the x-axis, and the next value indicates the position of the y-axis.
GroupBezierCurveFlag	This field, which is only present in the binary representation, indicates the presence of the GroupBezierCurve element. If it is set to "1," the GroupBezierCurve element follows.
GroupBezierCurve	It describes a hand contour or a hand trajectory in the form of a set of curves in 2-D space. A curve can be represented in the form of a group of 2-D coordinates.
CenterPositionFlag	This field, which is only present in the binary representation, indicates the presence of the CenterPosition element. If it is set to "1," the CenterPosition element follows.
NbrOfHandGestureFrame	This field, which is only present in the binary representation, describes the number of frames with hand gestures in the form of the 2-D coordinates
CenterPosition	The origin of the coordinate is the top-leftmost corner. It describes the centre position of the detected hand contour in the form of 2-D coordinates. A hand trajectory can be generated from a set of centre positions of hands later. The first value indicates the position of the x-axis, and the next value indicates the position of the y-axis
useLastContourFlag	This field, which is only present in the binary representation, indicates the presence of the useLastContour element. If it is set to "1," the useLastContour element follows.
useLastContour	It describes whether the current contour description is available or the last available contour is used. Suppose there is no need to generate a hand contour description of the detected hand for each frame or the valid hand contour is not detected for a particular frame. In that case, the last available contour description can be used instead of the current frame's contour description.

Name	Definition
GroupBezierCurveType	Tool for describing a set of Bezier curves description. A group of Bezier curve descriptions can be used to represent a hand gesture or a trajectory. Bezier curve is composed of two control points (P_1, P_2), one is a start point (P_0), and the other is an endpoint (P_3). Two control points are used for the curvature of contour (Figure 9).  **Figure 9 — Bezier curve**
InitialStartPoint	The origin of the coordinate is the top-leftmost corner. It describes the initial point for the first Bezier curve of Bezier curves representing the whole hand contour. By describing the initial point, the overlapping point between the consecutive curves can be described once instead of describing the same point redundantly. The start and endpoint of the hand contour called `InitialStartPoint` are the same due to the closed hand contour. `InitialStartPoint` is the same as the last Bezier curve's endpoint when a set of consecutive Bezer curves represents a hand contour. The first value indicates the position of the x-axis, and the next value indicates the position of the y-axis
BezierCurve	It describes a part of the hand contour in the form of a Bezier curve. For each Bezier curve (the order of Bezier curve n + 1), points are used to fitting the given curve by a polynomial. The n-points represent each curve of consecutive curves representing hand contour instead of (n + 1) points. Each point is described as a form of coordinate in 2-D space.
OrderOfBezierCurveFlag	This field, which is only present in the binary representation, indicates the presence of the `OrderOfBezierCurve` element. If it is set to "1," the `OrderOfBezierCurve` element follows.
OrderOfBezierCurve	It describes the order of the Bezier curve, which is the order of polynomial fitting the given curve. The n-th Bezier curve (n + 1) points in 2-D space, consisting of the starting point, endpoint, and (n-1) control points, represents the given curve.
FittingErrorFlag	This field, which is only present in the binary representation, indicates the presence of the `FittingError` element. If it is set to "1," the `FittingError` element follows.
FittingError	It describes the fitting error of the Bezier curve. The fitting error is used to represent the Bezier curve's accuracy based on the given hand gesture in optional.

Name	Definition
BezierCurveType	Tool for describing a Bezier curve. A Bezier curve uses (the order of Bezier curve n + 1) points to fit the given curve by a polynomial. The n-points represent each consecutive curve representing hand contour instead of (n + 1) points. Each point is described as a form of coordinate in 2-D space.
ControlPoint	The origin of the coordinate is the top-leftmost corner. It describes a control point for the Bezier curve in the form of a 2-D coordinate. The first value indicates the position of the x-axis, and the next value indicates the position of the y-axis
StartEndPoint	The origin of the coordinate is the top-leftmost corner. It describes the start and the endpoint of a Bezier curve in the form of a 2-D coordinate. The start and endpoint are the overlapping points between consecutive curves representing the given hand contour. Because one hand gesture consists of Bezier curves, each Bezier curve's start point and endpoint are overlapped. To avoid duplication of the start point and the endpoint, StartEndPoint is used for both points. The first value indicates the position of the x-axis, and the next value indicates the position of the y-axis
HandTrajectoryChoiceFlag	This field, which is only present in the binary representation, indicates the HandTrajectory element or HandCountour element. If it is set to "1," the HandTrajectory element follows; otherwise, the HandCountour element follows.
HandTrajectoryType	Tool for describing a hand trajectory. A hand trajectory description can represent the general hand trajectory, from which gesture-based interaction commands can be generated due to gesture recognition. It can be represented by a group of Bezier curves or a sequence of centre positions by hands.
trajectoryTypeFlag	This field, which is only present in the binary representation, indicates the presence of the trajectoryType element. If it is set to "1," the trajectoryType element follows.
trajectoryType	It describes whether the trajectory is defined and generated in a predefined time interval unit or the moving duration. The type of motion interval (motionInterval) means the trajectory is generated during the hand moving duration identified by motion detection. The time interval (timeInterval) indicates the trajectory generated in the predefined time interval unit. true: time interval / false: motion interval

8.5.5 Example

In this instance, the right hand is used for generating a hand gesture command because the lefthand value is "false." A Bezier points list represents the set of Bezier curves comprising the hand contour. The hand contour generated by this example is shown in Figure 10.

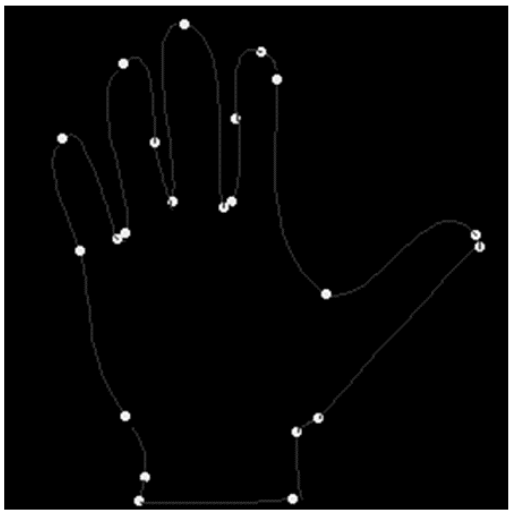


Figure 10 — Hand contour generated from an IoMT hand gesture detector

```
<mtdl:analysedData xsi:type="maov:HandGestureType" leftHand="false">
  <maov:HandContour>
    <maov:GroupBezierCurve>
      <maov:InitialStartPoint>80 22</maov:InitialStartPoint>
        <maov:BezierCurve>
          <maov:StartEndPoint>87 9</maov:StartEndPoint>
          <maov:ControlPoint>70 17</maov:ControlPoint>
          <maov:ControlPoint>90 82</maov:ControlPoint>
        </maov:BezierCurve>
        <maov:BezierCurve>
          <maov:StartEndPoint>85 103</maov:StartEndPoint>
          <maov:ControlPoint>84 103</maov:ControlPoint>
          <maov:ControlPoint>76 74</maov:ControlPoint>
        </maov:BezierCurve>
        <maov:BezierCurve>
          <maov:StartEndPoint>76 69</maov:StartEndPoint>
          <maov:ControlPoint>76 70</maov:ControlPoint>
          <maov:ControlPoint>78 6</maov:ControlPoint>
        </maov:BezierCurve>
        <maov:BezierCurve>
          <maov:StartEndPoint>57 33</maov:StartEndPoint>
          <maov:ControlPoint>40 52</maov:ControlPoint>
          <maov:ControlPoint>69 98</maov:ControlPoint>
        </maov:BezierCurve>
        <maov:BezierCurve>
          <maov:StartEndPoint>61 119</maov:StartEndPoint>
          <maov:ControlPoint>60 120</maov:ControlPoint>
          <maov:ControlPoint>57 118</maov:ControlPoint>
        </maov:BezierCurve>
        <maov:BezierCurve>
          <maov:StartEndPoint>27 71</maov:StartEndPoint>
          <maov:ControlPoint>18 85</maov:ControlPoint>
          <maov:ControlPoint>35 111</maov:ControlPoint>
        </maov:BezierCurve>
        <maov:BezierCurve>
          <maov:StartEndPoint>39 127</maov:StartEndPoint>
          <maov:ControlPoint>48 172</maov:ControlPoint>
          <maov:ControlPoint>38 174</maov:ControlPoint>
        </maov:BezierCurve>
```

```
<maov:BezierCurve>
  <maov:StartEndPoint>65 214</maov:StartEndPoint>
  <maov:ControlPoint>70 222</maov:ControlPoint>
  <maov:ControlPoint>53 230</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>71 241</maov:StartEndPoint>
  <maov:ControlPoint>70 241</maov:ControlPoint>
  <maov:ControlPoint>68 251</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>69 252</maov:StartEndPoint>
  <maov:ControlPoint>96 253</maov:ControlPoint>
  <maov:ControlPoint>123 253</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>151 250</maov:StartEndPoint>
  <maov:ControlPoint>149 250</maov:ControlPoint>
  <maov:ControlPoint>147 220</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>149 216</maov:StartEndPoint>
  <maov:ControlPoint>149 213</maov:ControlPoint>
  <maov:ControlPoint>158 210</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>160 209</maov:StartEndPoint>
  <maov:ControlPoint>162 260</maov:ControlPoint>
  <maov:ControlPoint>245 112</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>241 123</maov:StartEndPoint>
  <maov:ControlPoint>241 120</maov:ControlPoint>
  <maov:ControlPoint>240 117</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>239 116</maov:StartEndPoint>
  <maov:ControlPoint>215 87</maov:ControlPoint>
  <maov:ControlPoint>184 163</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>160 144</maov:StartEndPoint>
  <maov:ControlPoint>127 118</maov:ControlPoint>
  <maov:ControlPoint>140 68</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>138 32</maov:StartEndPoint>
  <maov:ControlPoint>137 27</maov:ControlPoint>
  <maov:ControlPoint>133 24</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>130 23</maov:StartEndPoint>
  <maov:ControlPoint>112 16</maov:ControlPoint>
  <maov:ControlPoint>118 54</maov:ControlPoint>
</maov:BezierCurve>
<maov:BezierCurve>
  <maov:StartEndPoint>118 58</maov:StartEndPoint>
  <maov:ControlPoint>118 41</maov:ControlPoint>
  <maov:ControlPoint>123 93</maov:ControlPoint>
</maov:BezierCurve>
```

```
        <maov:BezierCurve>
          <maov:StartEndPoint>114 101</maov:StartEndPoint>
          <maov:ControlPoint>113 101</maov:ControlPoint>
          <maov:ControlPoint>111 103</maov:ControlPoint>
        </maov:BezierCurve>
        <maov:BezierCurve>
          <maov:StartEndPoint>111 102</maov:StartEndPoint>
          <maov:ControlPoint>105 77</maov:ControlPoint>
          <maov:ControlPoint>118 9</maov:ControlPoint>
        </maov:BezierCurve>
      </maov:GroupBezierCurve>
  </maov:HandContour>
</mtdl:analysedData>
```

In this instance, users use their right hand to generate a hand trajectory because the `lefthand` value is "false." A list of Bezier points consisting of one trajectory representing the movement of the hand isis provided. The type of time interval samples the centre point list, so `trajectoryType` is "true." A hand trajectory generated from an IoMT hand gesture detector is shown in Figure 11.

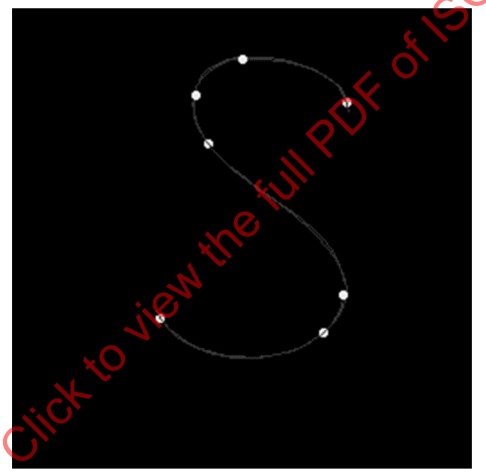


Figure 11 — Hand trajectory generated from an IoMT hand gesture detector

```
<mtdl:analysedData xsi:type="maov:HandGestureType" lefthand="false">
  <maov:HandTrajectory trajectoryType="true">
    <maov:GroupBezierCurve>
      <maov:InitialStartPoint>126 28</maov:InitialStartPoint>
      <maov:BezierCurve>
        <maov:StartEndPoint>126 28</maov:StartEndPoint>
        <maov:ControlPoint>99 32</maov:ControlPoint>
        <maov:ControlPoint>94 59</maov:ControlPoint>
      </maov:BezierCurve>
      <maov:BezierCurve>
        <maov:StartEndPoint>112 79</maov:StartEndPoint>
        <maov:ControlPoint>135 104</maov:ControlPoint>
        <maov:ControlPoint>226 148</maov:ControlPoint>
      </maov:BezierCurve>
      <maov:BezierCurve>
        <maov:StartEndPoint>165 186</maov:StartEndPoint>
        <maov:ControlPoint>149 195</maov:ControlPoint>
        <maov:ControlPoint>80 198</maov:ControlPoint>
      </maov:BezierCurve>
      <maov:BezierCurve>
        <maov:StartEndPoint>80 167</maov:StartEndPoint>
```

```
        <maov:ControlPoint>80 200</maov:ControlPoint>
        <maov:ControlPoint>188 207</maov:ControlPoint>
      </maov:BezierCurve>
      <maov:BezierCurve>
        <maov:StartEndPoint>185 153</maov:StartEndPoint>
        <maov:ControlPoint>182 104</maov:ControlPoint>
        <maov:ControlPoint>85 97</maov:ControlPoint>
      </maov:BezierCurve>
      <maov:BezierCurve>
        <maov:StartEndPoint>104 41</maov:StartEndPoint>
        <maov:ControlPoint>112 16</maov:ControlPoint>
        <maov:ControlPoint>187 27</maov:ControlPoint>
      </maov:BezierCurve>
      <maov:BezierCurve>
        <maov:StartEndPoint>187 57</maov:StartEndPoint>
        <maov:ControlPoint>187 30</maov:ControlPoint>
        <maov:ControlPoint>142 28</maov:ControlPoint>
      </maov:BezierCurve>
    </maov:GroupBezierCurve>
  </maov:HandTrajectory>
</mtdl:analysedData>
```

8.6 IoMT hand gesture recogniser

8.6.1 General

This subclause specifies the data format for a hand gesture recogniser output. The `MHandGestureRecogniser` obtains video sources including hand gestures from `MCameras` The `HandPostureType` represents a recognised hand posture.

8.6.2 Syntax

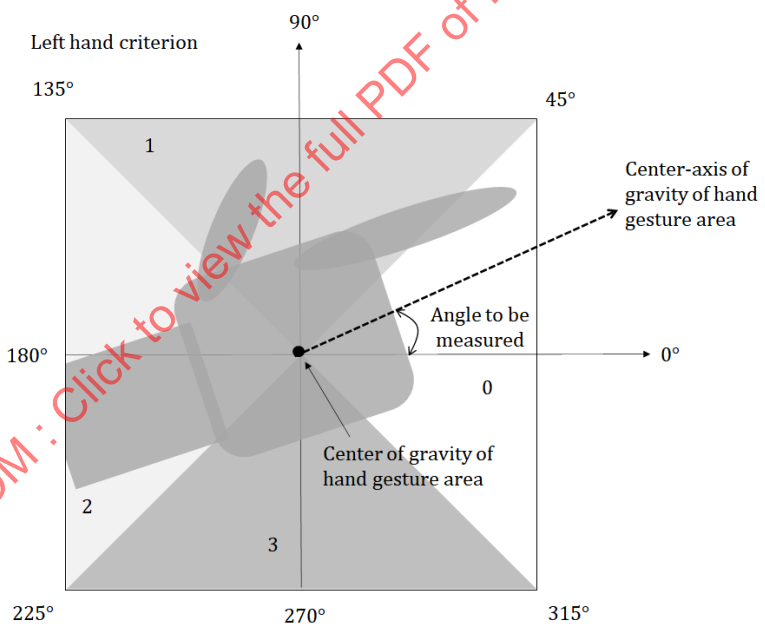
```
<!-- ################################################ -->
<!-- Definition of Hand Posture Type                  -->
<!-- ################################################ -->
<complexType name="HandPostureType">
   <complexContent>
      <extension base="mtdl:AnalysedDataBaseType">
         <attribute name="Thumb" type="boolean" use="required"/>
         <attribute name="Index" type="boolean" use="required"/>
         <attribute name="Middle" type="boolean" use="required"/>
         <attribute name="Ring" type="boolean" use="required"/>
         <attribute name="Little" type="boolean" use="required"/>
         <attribute name="Angle" default="0">
            <simpleType>
               <restriction base="integer">
                  <minInclusive value="0"/>
                  <maxInclusive value="3"/>
               </restriction>
            </simpleType>
         </attribute>
         <attribute name="InnerContour" type="boolean" use="required"/>
         <attribute name="HandSide" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

8.6.3 Binary Representation

HandPostureType {	Number of bits	Mnemonic
AnalysedDataBaseType		AnalysedDataBaseType
Thumb	1	bslbf
Index	1	bslbf
Middle	1	bslbf
Ring	1	bslbf
Little	1	bslbf
InnerContour	1	bslbf
HandSide	1	bslbf
}		

8.6.4 Semantics

Name	Definition
`HandPostureType`	It provides an abstract of Hand Posture Type.
`Thumb`	It describes a thumb finger status. If the finger is opened, then the value is true. Otherwise, the value is false.
`Index`	It describes an index finger status. If the finger is opened, then the value is true. Otherwise, the value is false.
`Middle`	It describes a middle finger status. If the finger is opened, then the value is true. Otherwise, the value is false.
`Ring`	It describes a ring finger status. If the finger is opened, then the value is true. Otherwise, the value is false.
`Little`	It describes a little finger status. If the finger is opened, then the value is true. Otherwise, the value is false.

Name	Definition
Angle	It describes an angle of a hand gesture representing the counterclockwise rotation in the unit of 90°. The centre of gravity of the hand area is placed at the origin of the 2D plane. The angle between the centre axis of gravity of the hand gesture area and the positive horizontal line (zero degrees) is measured as shown in Figure 12. The start point and endpoint of the centre axis of gravity is the centre of gravity and the figure side of the hand, respectively. In this way, zero degrees is when the centre axis of gravity is placed in the positive horizontal line, and the degree is increased counterclockwise. The measured angle is quantized into one of the four regions as shown in Figure 12, then the value of the element of angle represents the corresponding region below. 0: 315 °~45° (white) / 1: 45°~135° (blue) / 2: 135°~215° (green) / 3: 215°~315° (yellow)  **Figure 12 — An angle description of a hand gesture**
InnerContour	It describes whether an inner contour exists or not in the given hand contour. If the value of `Innercontour` is “true,” the given hand contour indicates the gesture of “okay.”
HandSide	It describes the hand side from the user’s point of view. If “true,” then palm-side of hand, otherwise, the back-side of hand.

8.6.5 Example

Values are “true,” the thumb and index fingers are open, and the remaining fingers, in which values are “false,” are closed. `Angle` 0 means the hand posture where the palm side is located between 315°~45°. And there is no inner contour because the `InnerContour value` is “false.” Because the `HandSide` value is “true,” the palm side is in the user’s point of view.

```
<mtdl:analysedData xsi:type="maov:HandPostureType" Thumb="true" Index="true"
Middle="false" Ring="false" Little="false" Angle="0" InnerContour="false"
HandSide="true"/>
```

8.7 IoMT hand gesture command generator

8.7.1 General

This subclause specifies a data format for a hand gesture command generator output. The `MHandGestureCommandGenerator` obtains a hand gesture recogniser. The `HandGestureCommandType` is used to represent information of a hand gesture command generated.

8.7.2 Syntax

```
<!-- ############################################## -->
<!-- Definition of Hand Gesture Command Generator Type -->
<!-- ############################################## -->
<complexType name="HandGestureCommandType">
   <complexContent>
      <extension base="mtdl:AnalysedDataBaseType">
         <sequence>
            <element name="HandGestureCommand" type="mpeg7:termReferenceType"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

8.7.3 Binary Representation

HandGestureCommandType {	**Number of bits**	**Mnemonic**
AnalysedDataBaseType		AnalysedDataBaseType
HandGestureCommand	8	bslbf
}		

8.7.4 Semantics

Name	Definition
HandGestureCommandType	Tool for describing an output of a hand gesture command generator.
HandGestureCommand	It describes the capability parameters of the hand gesture command generator. The type of capability shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the GestureCommandCS defined in A.4.3 (Table A.16).

8.7.5 Example

In this instance, the Recognised hand posture in the hand gesture recogniser is mapped into corresponding specific command. In this example, the mapped command is "PLAY" something.

```
<mtdl:analysedData xsi:type="maov:HandGestureCommandType">
   <maov:HandGestureCommand>urn:mpeg:mpeg-IoMT:01-GestureCommandCS-
NS:play</maov:HandGestureCommand>
</mtdl:analysedData>
```

8.8 IoMT healthcare information generator

8.8.1 General

This subclause specifies the data format for a healthcare information generator output. The MHealthcareInformationGenerator obtains patient records of the hospital. The HealthcareInfoType is used to represent information of healthcare information generated.

8.8.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of Healthcare Information Type        -->
<!-- ################################################ -->
<complexType name="HealthcareInfoType">
<complexContent>
   <extension base="mtdl:AnalysedDataBaseType">
      <sequence>
         <element name="Patient" type="maov:PatientType"/>
         <element name="CommonMediaInfo" type="maov:CommonMediaInfoType"/>
         <choice>
            <element name="ImageInfo" type="maov:ImageInfoType"/>
            <element name="VideoInfo" type="maov:VideoInfoType"/>
         </choice>
      </sequence>
   </extension>
</complexContent>
</complexType>

<complexType name="PatientType">
```

```
    <all>
        <element name="PatientName" type="maov:PatientNameType" minOccurs="0"/>
        <element name="PatientAge" type="string" minOccurs="0"/>
        <element name="PatientSex" type="string" minOccurs="0"/>
    </all>
</complexType>

<complexType name="PatientNameType">
    <all>
        <element name="FamilyName" type="string" minOccurs="0"/>
        <element name="GivenName" type="string" minOccurs="0"/>
        <element name="MiddleName" type="string" minOccurs="0"/>
    </all>
</complexType>

<complexType name="CommonMediaInfoType">
    <all>
        <element name="BitDepth" type="decimal" minOccurs="0"/>
        <element name="Width" type="decimal" minOccurs="0"/>
        <element name="Height" type="decimal" minOccurs="0"/>
        <element name="SamplesPerPixel" type="decimal" minOccurs="0"/>
        <element name="PixelAspectRatioX" type="decimal" minOccurs="0"/>
<element name="PixelAspectRatioY" type="decimal" minOccurs="0"/>
    </all>
</complexType>

<complexType name="ImageInfoType">
    <all>
        <element name="WindowWidth" type="decimal" minOccurs="0"/>
        <element name="WindowCenter" type="mpeg7:integerVector" minOccurs="0"/>
        <element name="NumberOfFrames" type="decimal" minOccurs="0"/>
    </all>
</complexType>

<complexType name="VideoInfoType">
    <all>
        <element name="VideoFormat" type="string" minOccurs="0"/>
        <element name="FrameRate" type="decimal" minOccurs="0"/>
    </all>
</complexType>
```


8.8.3 Binary Representation

HealthcareInfoType {	**Number of bits**	**Mnemonic**
AnalysedDataBaseType		AnalysedDataBaseType
ImageInfoChoiceFlag	1	bslbf
Patient		PatientType
CommonMediaInfo		CommonMediaInfoType
if (ImageInfoChoiceFlag) {		
ImageInfo		ImageInfoType
}		
Else {		
VideoInfo		VideoInfoType
}		
}		

PatientType {	Number of bits	Mnemonic
patientAgeFlag	1	bslbf
patientSexFlag	1	bslbf
patientNameFlag	1	bslbf
if (patientAgeFlag) {		
PatientAge	See ISO 10646	UTF-8
}		
if (patientSexFlag) {		
PatientSex	See ISO 10646	UTF-8
}		
if (patientNameFlag) {		
PatientName		PatientNameType
}		
}		

PatientNameType {	Number of bits	Mnemonic
familyNameFlag	1	bslbf
givenNameFlag	1	bslbf
middleNameFlag	1	bslbf
if (familyNameFlag) {		
FamilyName	See ISO 10646	UTF-8
}		
if (givenNameFlag) {		
GivenName	See ISO 10646	UTF-8
}		
if (middleNameFlag) {		
MiddleName	See ISO 10646	UTF-8
}		
}		

CommonMediaInfoType {	Number of bits	Mnemonic
bitDepthFlag	1	bslbf
widthFlag	1	bslbf
heightFlag	1	bslbf
samplesPerPixelFlag	1	bslbf
pixelAspectRatioXFlag	1	bslbf
pixelAspectRatioYFlag	1	bslbf
if (bitDepthFlag) {		
BitDepth	8	uimsbf
}		
if (widthFlag) {		
Width	16	uimsbf
}		
if (heightFlag) {		
Height	16	uimsbf
}		

CommonMediaInfoType {	Number of bits	Mnemonic
if (samplesPerPixelFlag) {		
SamplesPerPixel	8	uimsbf
}		
if (pixelAspectRatioXFlag) {		
PixelAspectRatioX	8	uimsbf
}		
if (pixelAspectRatioYFlag) {		
PixelAspectRatioY	8	uimsbf
}		
}		

ImageInfoType {	Number of bits	Mnemonic
windowWidthFlag	1	bslbf
windowCenterFlag	1	bslbf
numberOfFramesFlag	1	bslbf
if (windowWidthFlag) {		
WindowWidth	16	uimsbf
}		
if (windowCenterFlag) {		
WindowCenter	16*2	uimsbf
}		
if (numberOfFramesFlag) {		
NumberOfFrames	16	uimsbf
}		
}		

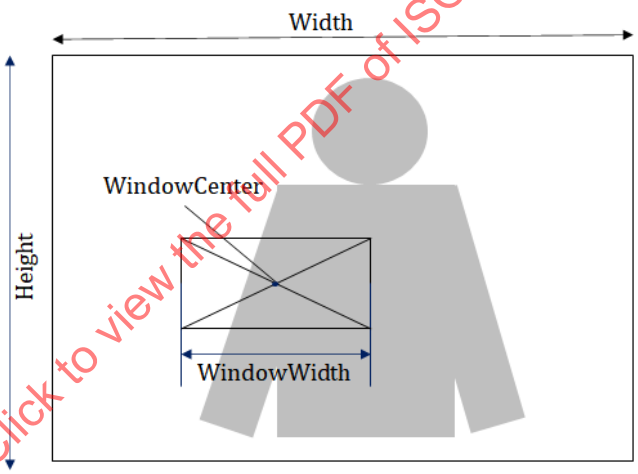
VideoInfoType {	Number of bits	Mnemonic
videoFormatFlag	1	bslbf
frameRateFlag	1	bslbf
if (videoFormatFlag) {		
VideoFormat	See ISO 10646	UTF-8
}		
if (frameRateFlag) {		
FrameRate	8	uimsbf
}		
}		

8.8.4 Semantics

Name	Definition
`HealthcareInfoType`	Tool for describing the healthcare information related to a patient and image/video characteristics.
`Patient`	It describes the patient-related information which is obtained from a speech recogniser.

Name	Definition
CommonMediaInfo	It describes the common media-related information on videos and images obtained from image/video characteristics and/or is extracted from the incoming image/video.
ImageInfoChoiceFlag	This field, which is only present in the binary representation, indicates the presence of the ImageInfo element or VideoInfo element. If it is set to "1," the ImageInfo element follows; otherwise, the VideoInfo element follows.
ImageInfo	It describes the image information obtained from a hand gesture recogniser and/or is extracted from the incoming image.
VideoInfo	It describes the video information obtained from video characteristics and/or is extracted from the incoming video.
patientNameFlag	This field, which is only present in the binary representation, indicates the presence of the PatientName element. If it is set to "1," the PatientName element follows.
patientAgeFlag	This field, which is only present in the binary representation, indicates the presence of the PatientAge element. If it is set to "1," the PatientAge element follows.
patientSexFlag	This field, which is only present in the binary representation, indicates the presence of the PatientSex element. If it is set to "1," the PatientSex element follows.
PatientName	It describes the full name of a patient, including family name, given name, and middle name.
PatientAge	A patient's age.
PatientSex	A patient's sex.
PatientNameType	It provides the information of the patient name.
familyNameFlag	This field, which is only present in the binary representation, indicates the presence of the FamilyName element. If it is set to "1," the FamilyName element follows.
givenNameFlag	This field, which is only present in the binary representation, indicates the presence of the GivenName element. If it is set to "1," the GivenName element follows.
middleNameFlag	This field, which is only present in the binary representation, indicates the presence of the MiddleName element. If it is set to "1," the MiddleName element follows.
FamilyName	A patient's family name.
GivenName	A patient's given name.
MiddleName	A patient's middle name.

Name	Definition
CommonMediaInfoType	Tool for describing the common information on image/video characteristics.
bitDepthFlag	This field, which is only present in the binary representation, indicates the presence of the BitDepth element. If it is set to "1," the BitDepth element follows.
widthFlag	This field, which is only present in the binary representation, indicates the presence of the Width element. If it is set to "1," the Width element follows.
heightFlag	This field, which is only present in the binary representation, indicates the presence of the Height element. If it is set to "1," the Height element follows.
samplesPerPixelFlag	This field, which is only present in the binary representation, indicates the presence of the SamplesPerPixel element. If it is set to "1," the SamplesPerPixel element follows.
pixelAspectRatioXFlag	This field, which is only present in the binary representation, indicates the presence of the PixelAspectRatioX element. If it is set to "1," the PixelAspectRatioX element follows.
pixelAspectRatioYFlag	This field, which is only present in the binary representation, indicates the presence of the PixelAspectRatioY element. If it is set to "1," the PixelAspectRatioY element follows.
BitDepth	It describes the number of bits allocated to each sample of the pixel. Each sample shall have the same number of bits to represent its value.
Width	It describes the width of the image/video in the unit of the pixel.
Height	It describes the height of the image/video in the unit of the pixel.
SamplesPerPixel	It describes the number of colour components per pixel. For monochrome image/video, the number of components is 1. For RGB and other three-colour models such as YUV and YCbCr, the value of this element is 3.
PixelAspectRatioX	It describes the horizontal size ratio of the pixels in the image specified by an integer value.
PixelAspectRatioY	It describes the vertical size ratio of the pixels in the image specified by an integer value.
ImageInfoType	It provides abstract information on image characteristics.
windowWidthFlag	This field, which is only present in the binary representation, indicates the presence of the WindowWidth element. If it is set to "1," the WindowWidth element follows.

Name	Definition
windowCenterFlag	This field, which is only present in the binary representation, indicates the presence of the WindowCenter element. If it is set to "1," the WindowCenter element follows.
numberOfFramesFlag	This field, which is only present in the binary representation, indicates the presence of the NumberOfFrames element. If it is set to "1," the NumberOfFrames element follows.
WindowWidth	Window width specifies the width of the image to be displayed in the unit of the pixel.
WindowCenter	The WindowCenter specifies the image's centre position as a 2D coordinate (Figure 13). The position of (0, 0) is the top-left pixel position in an image. The first value represents X coordination, and the last value represents Y coordination.  **Figure 13 — A window centre**
NumberOfFrames	The number of frames in a multi-frame image is a series of images with a common image header. The value of this element is generated by a speech recogniser.
VideoInfoType	It provides abstract information on video characteristics.
videoFormatFlag	This field, which is only present in the binary representation, indicates the presence of the VideoFormat element. If it is set to "1," the VideoFormat element follows.
frameRateFlag	This field, which is only present in the binary representation, indicates the presence of the FrameRate element. If it is set to "1," the FrameRate element follows.
VideoFormat	It describes the compression format of a video.
FrameRate	It describes the number of frames per second.

8.8.5 Examples

In this instance, a patient's family name is `Chun`, and his given name is `Sungmoon`. The gender of the patient is `male`. He is `40` years old. This example presents information about the image. It needs an `8-bit` depth in which the pixel value is available in the range of 0 ~ 255. The image resolution with the width by height is `1920x1080`. Pixel aspect ratio is `1:1`, which means a square pixel. The `WindowWidth` and the `WindowCenter` indicate a capturing region where the width is `1000` pixels, and the centre position is `(500, 250)`. And only one picture is ready to be transmitted.

Example A: image

```
<mtdl:analysedData xsi:type="maov:HealthcareInfoType">
  <maov:Patient>
    <maov:PatientName>
      <maov:FamilyName>Chun</maov:FamilyName>
      <maov:GivenName>Sungmoon</maov:GivenName>
    </maov:PatientName>
      <maov:PatientAge>40</maov:PatientAge>
      <maov:PatientSex>Male</maov:PatientSex>
  </maov:Patient>
  <maov:CommonMediaInfo>
    <maov:BitDepth>8</maov:BitDepth>
    <maov:Width>1920</maov:Width>
    <maov:Height>1080</maov:Height>
    <maov:SamplesPerPixel>3</maov:SamplesPerPixel>
    <maov:PixelAspectRatioX>1</maov:PixelAspectRatioX>
    <maov:PixelAspectRatioY>1</maov:PixelAspectRatioY>
  </maov:CommonMediaInfo>
  <maov:ImageInfo>
    <maov:WindowWidth>1000</maov:WindowWidth>
    <maov:WindowCenter>500 250</maov:WindowCenter>
    <maov:NumberOfFrames>1</maov:NumberOfFrames>
  </maov:ImageInfo>
</mtdl:analysedData>
```

In this instance, a patient's family name is `Chun`, and his given name is `Sungmoon`. The gender of the patient is `male`. He is `40` years old. This example presents information about the `image`. It needs an `8-bit` depth in which the pixel values is available in the range of 0 ~ 255. The image resolution with the width by height is `1920x1080`. Pixel aspect ratio is `1:1`, which means a square pixel. This video is compressed by using `MPEG-4 AVC`. The frame rate is `60Hz`.

Example B: video

```
<mtdl:analysedData xsi:type="maov:HealthcareInfoType">
  <maov:Patient>
    <maov:PatientName>
      <maov:FamilyName>Chun</maov:FamilyName>
      <maov:GivenName>Sungmoon</maov:GivenName>
    </maov:PatientName>
    <maov:PatientAge>40</maov:PatientAge>
    <maov:PatientSex>Male</maov:PatientSex>
  </maov:Patient>
  <maov:CommonMediaInfo>
    <maov:BitDepth>8</maov:BitDepth>
    <maov:Width>1920</maov:Width>
    <maov:Height>1080</maov:Height>
```

```
    <maov:SamplesPerPixel>3</maov:SamplesPerPixel>
    <maov:PixelAspectRatioX>1</maov:PixelAspectRatioX>
    <maov:PixelAspectRatioY>1</maov:PixelAspectRatioY>
  </maov:CommonMediaInfo>
  <maov:VideoInfo>
    <maov:VideoFormat>MPEG-4 AVC</maov:VideoFormat>
    <maov:FrameRate>60</maov:FrameRate>
  </maov:VideoInfo>
</mtdl:analysedData>
```

8.9 IoMT odour image to scent converter

8.9.1 General

This subclause specifies the data format for an odour image to scent converter output. The `MOdourImageToScentConverter` obtains odour images from `MCamera` and recognises the scent effect label and characteristic with confidence. The `odourImageType` is used to represent the recognised odour image.

8.9.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of Odour Image to Scent Converter     -->
<!-- ################################################ -->
<complexType name="odourImageType">
   <complexContent>
      <extension base="mtdl:AnalysedDataBaseType">
         <sequence>
            <element name="odourImageOutputList"
type="maov:odourImageOutputListType"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<complexType name="odourImageOutputListType">
   <sequence>
      <element name="odourImageOutput" type="maov:odourImageOutputType"
maxOccurs="unbounded"/>
   </sequence>
</complexType>

<complexType name="odourImageOutputType">
   <attribute name="odourImageLabel" type="mpeg7:termReferenceType"
use="optional"/>
   <attribute name="odourImageCharacteristics" type="mpeg7:termReferenceType"
use="optional"/>
   <attribute name="confidenceLevel" type="unsignedInt" use="optional"/>
   <attribute name="confidenceLevelofCharacterlistics" type="unsignedInt"
use="optional"/>
</complexType>
```

8.9.3 Binary Representation

odourImageType {	Number of bits	Mnemonic
AnalysedDataBaseType		AnalysedDataBaseType
odourImageOutputList		odourImageOutputListType
}		

odourImageOutputListType {	Number of bits	Mnemonic
NbrOfImageOutput	8	uimsbf
For (i=0; i< NbrOfImageOutput; i++) {		
odourImageOutput[i]		odourImageOutputType
}		
}		

odourImageOutputType {	Number of bits	Mnemonic
odourImageLabelFlag	1	bslbf
odourImageCharacteristicsFlag	1	bslbf
confidenceLevelFlag	1	bslbf
confidenceLevelofCharacteristicsFlag	1	bslbf
if(odourImageLabelFlag) {		
odourImageLabel	16	uimsbf
}		
if(odourImageCharacteristicsFlag) {		
odourImageCharacteristics	16	uimsbf
}		
if(confidenceLevelFlag) {		
confidenceLevel	8	uimsbf
}		
if(confidenceLevelofCharacteristicsFlag) {		
confidenceLevelofCharacteristics	8	uimsbf
}		
}		

8.9.4 Semantics

Name	Definition
`odourImageType`	Tool for describing the output of the "IoMT odour image to scent converter."
`odourImageOutputList`	It describes the output list of the "IoMT odour image to scent converter."
`odourImageOutputListType`	Tool for describing the output list of the "IoMT odour image to scent converter."
`odourImageOutput`	It describes the output of the "IoMT odour image to scent converter."

Name	Definition
NbrOfImageOutput	This field, only present in the binary representation, describes the number of image outputs with odour image labels produced by the odour image to scent converter.
odourImageOutputType	Tool for describing the scent effect in the odour image Recognised by the "IoMT odour image to scent converter."
odourImageLabelFlag	This field, which is only present in the binary representation, signals the presence of the attribute "odourImageLabel." A value of "1" means the attribute shall be used, and "0" means the attribute shall not be used
odrImageCharacteristics Flag	This field, which is only present in the binary representation, indicates the presence of the attribute "odourImageCharacteristics." A value of "1" means the attribute shall be used, and "0" means the attribute shall not be used
confidenceLevelFlag	This field, which is only present in the binary representation, signals the presence of the attribute "confidenceLevel." A value of "1" means the attribute shall be used, and "0" means the attribute shall not be used
confidenceLevelofCharacte risticsFlag	This field, which is only present in the binary representation, indicates the presence of the attribute "confidenceLevelofCharacteristics." A value of "1" means the attribute shall be used, and "0" means the attribute shall not be used
odourImageLabel	It describes the Recognised scent effect's label in the odour image. The type shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the ScentCS defined in ISO/IEC 23005-6.
odourImageCharacteristics	It describes the Recognised scent effect's characteristics in the odour image. The type shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the OdourImageCharacteristicsCS defined in A.4.1 (Table A.14).
confidenceLevel	It describes the confidence level of the scent effect's label Recognised by the "IoMT odour image to scent converter."
confidenceLevelofCharacte rlistics	It describes the confidence level of the scent effect's characteristics recognised by the "IoMT odour image to scent converter."

8.9.5 Example

This example shows the analysis result of the "IoMT odour image to scent converter." The result consists of two outputs with four attributes: odourImageLabel, odourImageCharacteritics, confidenceLevel and confidenceLevelCharacteristics. The first output of the result is odourImageLabel "coffee_cream," odourImageCharacteristics "DRY," confidenceLevel "60" and confidenceLevelCharacteristics "50." The second output of the result is odourImageLabel "orange" and confidenceLevel "0."

```
<mtdl:analysedData xsi:type="maov:odourImageType">
    <maov:odourImageOutputList>
        <maov:odourImageOutput odourImageLabel="urn:mpeg:mpeg-v:01-SI-ScentCS-
NS:coffee_cream" odourImageCharacteristics="urn:mpeg:mpeg-IoMT:01-
OdourImageCharacteristicsCS-NS:DRY"
        confidenceLevel="60" confidenceLevelofCharacterlistics="50"/>
        <maov:odourImageOutput odourImageLabel="urn:mpeg:mpeg-v:01-SI-ScentCS-
NS:orange" confidenceLevel="0"/>
    </maov:odourImageOutputList>
</mtdl:analysedData>
```

8.10 IoMT question analyser

8.10.1 General

This subclause specifies data formats to describe the outputs that the IoMT `QuestionAnalyser` can produce. The `QuestionAnalyser` takes a user's question in the text as an input and produces a question analysis result. The result of question analysis can generate user interaction commands and input to the Question Answering (QA) server to provide answers/information to the user. The `UserQuestionType` represents the analysed user's question, for which the QA service can provide answers. If it is analysed as a control command, it is sent to the actuator.

8.10.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of Question Analysis Type             -->
<!-- ################################################ -->
<complexType name="QuestionAnalysisType">
    <complexContent>
        <extension base="mtdl:AnalysedDataBaseType">
            <sequence>
                <element name="analysedQuestion" type="maov:UserQuestionType"/>
                <element name="language" type="language" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="UserQuestionType">
    <sequence>
        <element name="qtopic" type="string" minOccurs="0"/>
        <element name="qfocus" type="mpeg7:termReferenceType" minOccurs="0"/>
        <element name="qCsemantic" type="mpeg7:termReferenceType" minOccurs="0"/>
    </sequence>
    <attribute name="qdomain" type="string"/>
</complexType>
```

8.10.3 Binary Representation

QuestionAnalysisType {	Number of bits	Mnemonic
languageFlag	1	bslbf
AnalysedDataBaseType		AnalysedDataBaseType
analysedQuestion		UserQuestionType
If(languageFlag) {		
language	See ISO 10646	UTF-8
}		
}		

UserQuestionType {	Number of bits	Mnemonic
qtopicFlag	1	bslbf
qfocusFlag	1	bslbf
qCsemanticFlag	1	bslbf
If(qtopicFlag) {		
qtopic	See ISO 10646	UTF-8
}		
If(qfocusFlag) {		
qfocus	8	bslbf
}		
If(qCsemanticFlag) {		
qCsemantic	8	bslbf
}		
qdomain	See ISO 10646	UTF-8
}		

8.10.4 Semantics

Name	Definition
`QuestionAnalysisType`	It provides an abstract of the description of question analysis, which can be done in a processing unit.
`languageFlag`	This field, which is only present in the binary representation, indicates the presence of `language` element. If it is set to "1," the `language` element follows.
`anlysedQuestion`	It describes the analysed question resulting from the question analysis.
`language`	It indicates the language of the input question. If present, the `Language` element should take precedence over other language indications present within the input question.
`UserQuestionType`	It provides abstracts of the User Question description. It describes the user's utterance that is the output of the speech recognition process. User Question is sent to the QA server for providing answers to the user. If it is a control command, it is sent to the actuator.

Name	Definition
qtopicFlag	This field, which is only present in the binary representation, indicates the presence of the qtopic element. If it is set to "1," the qtopic element follows.
qtopic	It describes the topic of the question. Question topic is the object or event that the question asks. Ex. The question topic is "King Lear" in "Who is the author of King Lear?."
qfocusFlag	This field, which is only present in the binary representation, indicates the presence of the qfocus element. If it is set to "1," the qfocus element follows.
qfocus	It describes the focus of the question, which is one of 5W1H. The type of qfocus shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme used for this purpose is the QfocusCS defined in A.4.4 (Table A.17). Ex. What, where, who, what policy.
qCsemanticFlag	This field, which is only present in the binary representation, indicates the presence of the qCsemantic element. If it is set to "1," the qCsemantic element follows.
qCsemantic	It describes the question classification based on the meaning/purpose of the question. The type of question classification shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the QCsemanticCS defined in A.4.5 (Table A.18). Ex. What does MPEG stand for? (Request for terminology). Could you please turn on the TV? (Request for a command)
qdomain	It describes the domain of the question such as "science," "weather," "history." Ex. Who is the third king of the Yi dynasty in Korea? (qdomain: history)

8.10.5 Examples

This example shows the question analysis result of the user's question, "Who is the author of King Lear?" The result consists of analysedQuestion and language, "en-us." The question analysis result in the example shows that the domain of the question is "Literature," the topic of the question is "King Lear," the focus of the question is "WHO_QUESTION" and the purpose of the question is to "REQUEST_FOR_INFORMATION"

```
<mtdl:analysedData xsi:type="maov:QuestionAnalysisType">
    <maov:analysedQuestion qdomain="Literature">
        <maov:qtopic>urn:mpeg:mpeg-IoMT:01-QtopicCS-NS:King Lear</maov:qtopic>
        <maov:qfocus>urn:mpeg:mpeg-IoMT:01-QfocusCS-NS:WHO_QUESTION</maov:qfocus>
        <maov:qCsemantic>urn:mpeg:mpeg-IoMT:01-QCsemanticCS-
NS:REQUEST_FOR_INFORMATION</maov:qCsemantic>
    </maov:analysedQuestion>
    <maov:language>en-us</maov:language>
</mtdl:analysedData>
```

This 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_QUESTION" and the purpose of the question is to "REQUEST_FOR_METHOD."

```
<mtdl:analysedData xsi:type="maov:QuestionAnalysisType">
    <maov:analysedQuestion qdomain="Cooking">
        <maov:qtopic>urn:mpeg:mpeg-IoMT:01-QtopicCS-NS:Kimchi</maov:qtopic>
        <maov:qfocus>urn:mpeg:mpeg-IoMT:01-QfocusCS-NS:HOW_QUESTION</maov:qfocus>
        <maov:qCsemantic>urn:mpeg:mpeg-IoMT:01-QCsemanticCS-
NS:REQUEST_FOR_METHOD</maov:qCsemantic>
    </maov:analysedQuestion>
    <maov:language>en-us</maov:language>
</mtdl:analysedData>
```

8.11 IoMT music frequency analyser

8.11.1 General

This subclause specifies the data format for a Music Frequency Analyser output. The `MMusicFrequencyAnalyser` obtains audio source and sampling rate information from `MMicrophone` and calculates how many points are at a specific frequency amplitude. The `MusicCharacterType` is used to represent information about analysed music frequency.

8.11.2 Syntax

```
<!-- ##########################################-->
<!-- Definition of Analysed Music Frequency Type -->
<!-- ##########################################-->

<complexType name="AnalysedMusicFrequencyType">
<complexContent>
    <extension base="mtdl:AnalysedDataBaseType">
        <sequence>
            <element name="AudioSamplingRate" type="nonNegativeInteger"/>
            <element name="NumberOfPoints" type="nonNegativeInteger"/>
            <element name="Spectrogram" type="maov:SpectrogramType" maxOccurs="1"
minOccurs="1"/>
        </sequence>
    </extension>
</complexContent>
</complexType>

<complexType name="SpectrogramType">
    <sequence>
        <element name="SpectroPoint" type="maov:SpectroPointType"
maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="SpectroPointType">
    <sequence>
        <element name="FreqAmplitude" type="float"/>
        <element name="PointNumber" type="integer"/>
    </sequence>
</complexType>
```

8.11.3 Binary Representation

AnalysedMusicFrequencyType {	Number of bits	Mnemonic
AnalysedDataBaseType		AnalysedDataBaseType
AudioSamplingRate	24	uimsbf
NumberofPoints	16	uimsbf
Spectrogram		SpectrogramType
}		

SpectrogramType {	Number of bits	Mnemonic
SpectroPoint		SpectroPointType
}		

SpectroPointType {	Number of bits	Mnemonic
FreqAmplitude	32	flbf
PointNumber	16	uimsbf
}		

8.11.4 Semantics

Name	Definition
`AnalysedMusicFrequencyType`	Tool for describing analysed frequency information of music from `MMicrophone`.
`AudioSamplingRate`	It describes the audio sampling rate of the audio source.
`NumberOfPoints`	It describes the number of samples to be analysed at one time.
`Spectrogram`	It describes the spectrogram information of the audio source.
`SpectrogramType`	Tool for describing the result of the Fourier analysis of the signal or audio.
`SpectroPoint`	It describes the amplitude value and the count identifying a particular frequency which is corresponding to the amplitude.
`SpectroPointType`	Tool for describing the result of Fourier analysis result of the signal or audio.
`FreqAmplitude`	It describes the amplitude of a particular frequency of each point acquired by the Fourier analysis.
`PointNumber`	It describes the number identifying a particular frequency for which the `FreqAmplitude` is given.

8.11.5 Examples

This example shows that the sampling rate of an audio source is 44100 Hz, and the bandwidth will be 22050Hz by the Nyquist theorem. This given bandwidth will be divided into four bands, and the range of each band will be 5512.5Hz. The `FreqAmplitude` means the amplitude of each frequency band, so the `FreqAmplitude` of the first band is 337.5. The `FreqAmplitude` of the second band is 912.5, the `FreqAmplitude` of the third band is 33, and the `FreqAmplitude` of the last(highest) band is 12.5.

```
<mtdl:analysedData xsi:type="maov:AnalysedMusicFrequencyType">
    <maov:AudioSamplingRate>44100</maov:AudioSamplingRate>
    <maov:NumberOfPoints>8</maov:NumberOfPoints>
    <maov:Spectrogram>
        <maov:SpectroPoint>
            <maov:FreqAmplitude>337.5</maov:FreqAmplitude>
            <maov:PointNumber>0</maov:PointNumber>
        </maov:SpectroPoint>
        <maov:SpectroPoint>
            <maov:FreqAmplitude>912.5</maov:FreqAmplitude>
            <maov:PointNumber>1</maov:PointNumber>
        </maov:SpectroPoint>
        <maov:SpectroPoint>
            <maov:FreqAmplitude>33</maov:FreqAmplitude>
            <maov:PointNumber>2</maov:PointNumber>
        </maov:SpectroPoint>
        <maov:SpectroPoint>
            <maov:FreqAmplitude>12.5</maov:FreqAmplitude>
            <maov:PointNumber>3</maov:PointNumber>
        </maov:SpectroPoint>
    </maov:Spectrogram>
</mtdl:analysedData>
```

8.12 IoMT video content class generator

8.12.1 General

This subclause specifies data formats for a video content class generator output. The `MVideoContentClassGenerator` obtains a video source from `MCamera` or `MStorage`, analyses and generates a class of video content. The `VideoContentClassType` is used to represent information of analysed video content class.

8.12.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of Video Content Class Generator Type  -->
<!-- ################################################ -->

<complexType name="VideoContentClassType">
    <complexContent>
        <extension base="mtdl:AnalysedDataBaseType">
            <sequence>
                <element name="VideoContentClass" type="mpeg7:termReferenceType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

8.12.3 Binary Representation

VideoContentClassType {	Number of bits	Mnemonic
AnalysedDataBaseType		AnalysedDataBaseType
VideoContentClass	8	bsblf
}		

8.12.4 Semantics

Name	Definition
VideoContentClassType	Tool for describing the generated video content class.
VideoContentClass	It describes the analysed content class or genre of a video source. The name of the specific video class or genre shall be described using the mpeg7:termReferenceType defined in ISO/IEC 15938-5:2003, 7.6. A classification scheme that may be used for this purpose is the VideoContentClassCS defined in A.4.6 (Table A.19).

8.12.5 Examples

This example shows that generated video content class is sports which is analysed by MVideoContentClassGenerator.

```
<mtdl:analysedData xsi:type="maov:VideoContentClassType">
   <maov:VideoContentClass>urn:mpeg:mpeg-IoMT:01-VideoContentClassCS-
NS:Sports</maov:VideoContentClass>
</mtdl:analysedData>
```

8.13 IoMT face region detector

8.13.1 General

This subclause specifies data formats to describe the outputs that the MFaceRegionDetector can produce. The MFaceRegionDetector obtains a video source from MCamera and outputs a list of coordinates of faces in 2D image coordinates. The FaceRegionResultType is used to represent a list of information related to detected faces.

8.13.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of IoMT Face Region Detector Type     -->
<!-- ################################################ -->
<complexType name="FaceRegionResultType">
    <complexContent>
        <extension base="mtdl:AnalysedDataBaseType">
            <sequence>
                <element name="FaceRegionResultSet" minOccurs="0"
type="maov:FaceRegionSetType"/>
            </sequence>
```

```
        </extension>
      </complexContent>
</complexType>
<complexType name="FaceRegionSetType">
   <sequence>
       <element name="FaceRegion" type="maov:FaceRegionType" minOccurs="0"
maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="FaceRegionType">
   <sequence>
       <element name="x" type="unsignedInt" />
       <element name="y" type="unsignedInt" />
       <element name="w" type="unsignedInt" />
       <element name="h" type="unsignedInt" />
       <element name="FrameNumber" type="unsignedInt" />
   </sequence>
</complexType>
```

8.13.3 Binary Representation

FaceRegionResultType {	**Number of bits**	**Mnemonic**
AnalysedDataBaseType		AnalysedDataBaseType
FaceRegisonResultSetFlag	1	bslbf
iF (FaceRegionResultSetFlag){		
FaceRegionResultSet		FaceRegionSetType
}		
}		

FaceRegionSetType {	**Number of bits**	**Mnemonic**
NbrOfDetectedFaceRegion	*8*	uimsbf
FaceRegionFlag	1	bslbf
If (FaceRegionFlag){		
For(i=0; i<NbrOfDetectedFaceRegion; i++){		
FaceRegion[i]		FaceRegionType
}		
}		
}		

FaceRegionType {	**Number of bits**	**Mnemonic**
x	*16*	uimsbf
y	*16*	uimsbf
w	*16*	uimsbf
h	*16*	uimsbf
FrameNumber	*16*	uimsbf
}		

8.13.4 Semantics

Name	Definition
FaceRegionResultType	Tool for describing face region result
FaceRegionResultSetFlag	This field, which is only present in the binary representation, indicates the presence of the FaceRegionResultSet element. If it is set to "1," the FaceRegionResultSet element follows.
FaceRegionResultSet	It describes the detected set of face region(s)
FaceRegionResultSetType	Tool for describing a set of detected face region(s)
FaceRegionFlag	This field, which is only present in the binary representation, indicates the presence of the FaceRegion element. If it is set to "1," the FaceRegion element follows.
FaceRegion	It describes region information of detected faces.
FaceRegionType	Tool for describing the detected face region.
x	It describes the x location of the detected face. (top-left x coordinate)
y	It describes the y location of the detected face. (top-left y coordinate)
W	It describes the width of the bounding box of a detected face.
h	It describes the height of the bounding box of a detected face.
FrameNumber	When the format of the media source is a video stream, it can have sequence order information of decoded input stream. FrameNumber represents the frame order information of the input video stream in integer form.

8.13.5 Example

This example shows a detected face on (32, 48) with width=16, height = 16 at 5th frame of video.

```
<mtdl:analysedData xsi:type="maov:FaceRegionResultType">
      <maov:FaceRegionResultSet>
          <maov:FaceRegion>
              <maov:x>33</maov:x>
              <maov:y>52</maov:y>
              <maov:w>120</maov:w>
              <maov:h>110</maov:h>
              <maov:FrameNumber>5</maov:FrameNumber>
          </maov:FaceRegion>
      </maov:FaceRegionResultSet>
  </mtdl:analysedData>
```

8.14 IoMT face verifier

8.14.1 General

This subclause specifies data formats to describe the outputs that the `MFaceVerifier` can produce. The `MFaceVerifier` obtains a video source from `MCamera` and images compared to the `MThing` and outputs comparison results in a Boolean with a confidence score.

8.14.2 Syntax

```
<!-- ################################################ -->
<!-- Definition of IoMT Face Verifier Type            -->
<!-- ################################################ -->

<complexType name="FaceVerificationType">
       <complexContent>
          <extension base="mtdl:AnalysedDataBaseType">
          <sequence>
             <element name="FaceVerificationResult" minOccurs="1"
type="boolean"/>
             <element name="ConfidenceScore" minOccurs="0" maxOccurs="1">
                <simpleType>
                   <restriction base="float">
                      <minInclusive value="0.0"/>
                      <maxInclusive value="100.0"/>
                   </restriction>
                </simpleType>
             </element>
             </sequence>
          </extension>
       </complexContent>
   </complexType>
```

8.14.3 Binary Representation

FaceVerificationType {	Number of bits	Mnemonic
AnalysedDataBaseType		AnalysedDataBaseType
FaceVerificationResult	1	bslbf
ConfidenceResultFlag	1	bslbf
If (ConfidenceResultFlag)		
{		
ConfidenceScore	32	flbf
}		
}		

8.14.4 Semantics

Name	Definition
FaceVerificationType	Tool for describing the face verification result
FaceVerificationResult	It describes the verification result. false: Unauthorized face true: Authorized face
ConfidenceResultFlag	This field, which is only present in the binary representation, indicates the presence of the ConfidenceScore element. If it is set to"1," the ConfidenceScore element follows.
ConfidenceScore	It describes a confidence score of the verification result.

8.14.5 Example

This example shows a face verification result with TRUE and confidence score of 76.5

```
<mtdl:analysedData xsi:type="maov:FaceVerificationType">
   <maov:FaceVerificationResult>true</maov:FaceVerificationResult>
   <maov:ConfidenceScore>76.5</maov:ConfidenceScore>
</mtdl:analysedData>
```

8.15 IoMT security title generator

8.15.1 General

This subclause specifies data formats to describe the outputs that MSecurityTitleGenerator can produce. The MSecurityTitleGenerator monitors video from MCamera, and if MSecurityTitleGenerator detects some security event, MSecurityTitleGenerator generates security event information which are detection time, location (Latitude, Longitude), the detected security event title. The SecurityTitleGeneratorType is used to create the information of the analysed and generated security title.

8.15.2 Syntax

```
<!-- ########################################### -->
<!-- Definition of Security Title Generator Type -->
<!-- ########################################### -->

<complexType name="GeneratedSecurityTitleType">
<complexContent>
   <extension base="mtdl:AnalysedDataBaseType">
       <sequence>
      <element name="DetectedTime" type="mpeg7:timePointType"/>
      <element name="Latitude" type="float" minOccurs="0"/>
      <element name="Longitude" type="float" minOccurs="0"/>
      <element name="SecurityEvent" type="mpeg7:termReferenceType"/>
```

```
        </sequence>
    </extension>
</complexContent>
</complexType>
```

8.15.3 Binary Representation

GeneratedSecurityTitleType {	**Number of Bits**	**Mnemonic**
AnalysedDataBaseType		AnalysedDataBaseType
LatitudeFlag	1	bslbf
LongitudeFlag	1	bslbf
DetectedTime		mpeg7:TimePointType
if (LatitudeFlag)		
Latitude	32	fsfb
if (LongitudeFlag)		
Longitude	32	fsfb
SecurityEvent	8	bslbf
}		

8.15.4 Semantics

Name	**Definition**
`GeneratedSecurityTitleType`	A tool for describing the descriptors generated event-specific.
`DetectedTime`	It describes the captured time of the video source.
`LatitudeFlag`	This field, which is only present in the binary representation, indicates the presence of the `Latitude` element. If it is set to"1," the `Latitude` element follows.
`LongitudeFlag`	This field, which is only present in the binary representation, indicates the presence of the `Longitude` element. If it is set to"1," the `Longitude` element follows.
`Latitude`	It describes a signed latitude value.
`Longitude`	It describes a signed longitude value.
`SecurityEvent`	Represents the event detected by analysing the video source. The type of event shall be described using the `mpeg7:termReferenceType` defined in 7.6 of ISO/IEC 15938-5:2003. A classification scheme that may be used for this purpose is the `SecurityEventCS` defined in A.4.7 (Table A.20).

8.15.5 Example

This example shows the output of video analysis results and associated sensed data via a security title generator. The MSecurityTitleGenerator detected an assault event in location where latitude value is 37.541525 and longitude value is 127.070382 (Seoul, Republic of Korea) at 10:47 UTC+9, 7th, July, 2019.

```
<mtdl:analysedData xsi:type="maov:GeneratedSecurityTitleType">
    <maov:DetectedTime >2019-07-07T10:47+09:00</maov:DetectedTime >
    <maov:Latitude>41.40338</maov:Latitude>
    <maov:Longitude>2.17403</maov:Longitude>
    <maov:SecurityEvent>urn:mpeg:mpeg-IoMT:01-SecurityEventCS-
NS:ASSAULT</maov:SecurityEvent>
</mtdl:analysedData>
```

8.16 IoMT light colour converter

8.16.1 General

This subclause specifies data formats to describe the outputs that MLightColourConveter can produce. The MLightColourConveter is used to represent the lighting value information suitable for the mood of the audio source by acquiring the analysed music frequency Information from the MMusicFrequencyAnalyser.

8.16.2 Syntax

```
<!-- ########################################### -->
<!-- Definition of analysed colour light information Type -->
<!-- ########################################### -->

<complexType name="analysedColourLightInfoType">
<complexContent>
    <extension base="mtdl:AnalysedDataBaseType">
        <sequence>
            <element name="brightness>
                <simpleType>
                    <restriction base="float">
                        <minInclusive value="0.0"/>
                        <maxInclusive value="1.0"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="saturation">
                <simpleType>
                    <restriction base="float">
                        <minInclusive value="0.0"/>
                        <maxInclusive value="1.0"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="hue">
                <simpleType>
                    <restriction base="integer">
                        <minInclusive value="0"/>
                        <maxInclusive value="360"/>
```

```
                </restriction>
            </simpleType>
        </element>
    </sequence>
  </extension>
</complexContent>
</complexType>
```

8.16.3 Binary Representation

analysedColourLightInfoType{	**Number of Bits**	**Mnemonic**
AnalysedDataBaseType		AnalysedDataBaseType
brightness	32	fsfb
saturation	32	fsfb
hue	9	uimsbf
}		

8.16.4 Semantics

Name	**Definition**
analysedColourLightInfoType	Tool for describing the colour light information.
brightness	The brightness value of the colour suitable for the analysed mood of music by the light colour converter
saturation	The saturation value of the colour suitable for the analysed mood of music by the light colour converter
hue	The hue value of the colour suitable for the analysed mood of music by the light colour converter

8.16.5 Example

This example shows that analysed colour light information is the brightness 0.7, the saturation 0.8 and the hue 240.

```
<mtdl:analysedData xsi:type="maov:analysedColourLightInfoType">
    <maov:brightness>0.7</maov:brightness >
    <maov:saturation>0.8</maov:saturation >
    <maov:hue>240</maov:hue>
</mtdl:analysedData>
```

Annex A
(normative)

Classification scheme

A.1 MThingTypeCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-MThingTypeCS-NS">

<Term termId="MSensor">
    <Name xml:lang="en">Media Sensor</Name>
    <Definition xml:lang="en">
       A MThing capable of sensing media
    </Definition>
</Term>

<Term termId="MActuator">
    <Name xml:lang="en">Media Actuator</Name>
    <Definition xml:lang="en">
       A MThing capable of actuating, rendering, and/or displaying media and
related metadata
    </Definition>
</Term>

<Term termId="MAnalyser">
    <Name xml:lang="en">Media Analyser</Name>
    <Definition xml:lang="en">
       A MThing capable of analysing media, and producing related media and/or
metadata
    </Definition>
</Term>

<Term termId="MStorage">
    <Name xml:lang="en">Media Storage</Name>
    <Definition xml:lang="en">
       A MThing capable of storing media or metadata
    </Definition>
</Term>

<Term termId="MManager">
    <Name xml:lang="en">MThing Manager</Name>
    <Definition xml:lang="en">
       A MThing capable of managing a list of MThings
    </Definition>
</Term>

<Term termId="MAggregator">
    <Name xml:lang="en">MThing Aggregator</Name>
    <Definition xml:lang="en">
       A MThing composed of two or more sub-MThings
    </Definition>
</Term>
```

Table A.1 — Binary representation of `MThingTypeCS`

Binary representation	Term ID of MThingTypeCS
00000000	MSensor
00000001	MActuator
00000010	MAnalyser
00000011	MStorage
00000100	MManager
00000101	MAggregator
00000110 – 11111111	Reserved

A.2 CapabilityCS

A.2.1 SensorCapabilityCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-SensorCapabilityCS-NS">

<Term termId="SENSOR_DEFAULT">
    <Name xml:lang="en">DEFAULT</Name>
    <Definition xml:lang="en">
        Any MSensors
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_VIDEO">
    <Name xml:lang="en">CAPTURE_VIDEO</Name>
    <Definition xml:lang="en">
        An MSensor is capable of captureing a video
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_AUDIO">
    <Name xml:lang="en">CAPTURE_AUDIO</Name>
    <Definition xml:lang="en">
        An MSensor is capable of Capturing an audio
    </Definition>
</Term>

<Term termId="SENSOR_STREAM_VIDEO">
<Name xml:lang="en">STREAM_VIDEO</Name>
    <Definition xml:lang="en">
        An MSensor is capable of streaming a video
    </Definition>
</Term>

<Term termId="SENSOR_STREAM_STEREO_VIDEO">
<Name xml:lang="en">STREAM_STEREO_VIDEO</Name>
    <Definition xml:lang="en">
```

```
        An MSensor is capable of streaming a stereo video
    </Definition>
</Term>

<Term termId="SENSOR_STREAM_AUDIO">
    <Name xml:lang="en">STREAM_AUDIO</Name>
    <Definition xml:lang="en">
        An MSensor is capable of streaming an audio
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_DISTANCE">
    <Name xml:lang="en">MEASURE_DISTANCE</Name>
    <Definition xml:lang="en">
        An MSensor is capable of measuring a distance
    </Definition>
</Term>

  <Term termId="SENSOR_MEASURE_LATITUDE">
    <Name xml:lang="en">MEASURE_LATITUDE</Name>
    <Definition xml:lang="en">
        An MSensor is capable of measuring a latitude
    </Definition>
</Term>

  <Term termId="SENSOR_MEASURE_LONGITUDE">
    <Name xml:lang="en">MEASURE_LONGITUDE</Name>
    <Definition xml:lang="en">
        An MSensor is capable of measuring a longitude
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_AZIMUTH">
<Name xml:lang="en">MEASURE_AZIMUTH</Name>
   <Definition xml:lang="en">
        An MSensor is capable of measuring an azimuth
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_YAW_ANGLE">
    <Name xml:lang="en">MEASURE_YAW_ANGLE</Name>
    <Definition xml:lang="en">
        An MSensor is capable of measuring a pitch angle
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_PITCH_ANGLE">
    <Name xml:lang="en">MEASURE_PITCH_ANGLE</Name>
    <Definition xml:lang="en">
        An MSensor is capable of measuring a pitch angle
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ROLL_ANGLE">
    <Name xml:lang="en">MEASURE_ROLL_ANGLE</Name>
    <Definition xml:lang="en">
        An MSensor is capable of measuring a roll angle
    </Definition>
</Term>
```

```
<Term termId="SENSOR_CAPTURE_STEREO_VIDEO">
    <Name xml:lang="en">CAPTURE_STEREO_VIDEO</Name>
    <Definition xml:lang="en">
       An MSensor is capable of capturing a stereo video
</Term>

<Term termId="SENSOR_CAPTURE_IMAGE">
    <Name xml:lang="en">CAPTURE_IMAGE</Name>
    <Definition xml:lang="en">
       An MSensor is capable of capture an image
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_GLOBAL_POSITION">
    <Name xml:lang="en">MEASURE_GLOBAL_POSITION</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the global position
    </Definition>
</Term>

<Term termId="SENSOR_READ_RFID">
    <Name xml:lang="en">READ_RFID</Name>
    <Definition xml:lang="en">
      An Msensor is capable of reading RFID
    </Definition>
</Term>

<Term termId="SENSOR_DETECT_POSITION">
    <Name xml:lang="en">DETECT_POSITION</Name>
    <Definition xml:lang="en">
      An Msensor is capable of DETECTING POSITION
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_LIGHT_INTENSITY">
    <Name xml:lang="en">MEASURE_LIGHT_INTENSITY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING LIGHT INTENSITY
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_LIGHT_COLOUR">
    <Name xml:lang="en">MEASURE_LIGHT_COLOUR</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING LIGHT COLOUR
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_AMBIENT_NOISE">
    <Name xml:lang="en">MEASURE_AMBIENT_NOISE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING AMBIENT NOISE
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_TEMPERATURE">
    <Name xml:lang="en">MEASURE_TEMPERATURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING TEMPERATURE
    </Definition>
</Term>
```

```
<Term termId="SENSOR_MEASURE_HUMIDITY">
    <Name xml:lang="en">MEASURE_HUMIDITY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING HUMIDITY
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ATMOSPHERIC_PRESSURE">
    <Name xml:lang="en">MEASURE_ATMOSPHERIC_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING ATMOSPHERIC PRESSURE
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_PRESSURE">
    <Name xml:lang="en">MEASURE_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING PRESSURE
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_VELOCITY>
    <Name xml:lang="en">MEASURE_VELOCITY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING VELOCITY
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ACCELERATION>
    <Name xml:lang="en">MEASURE_ACCELERATION</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING ACCELERATION
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_AGULAR_ACCELERATION>
    <Name xml:lang="en">MEASURE_AGULAR_ACCELERATION</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING AGULAR ACCELERATION
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_AGULAR_VELOCITY>
    <Name xml:lang="en">MEASURE_AGULAR_VELOCITY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING AGULAR VELOCITY
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_FORCE>
    <Name xml:lang="en">MEASURE_FORCE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING FORCE
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_TORQUE>
    <Name xml:lang="en">MEASURE_TORQUE</Name>
    <Definition xml:lang="en">
```

```
      An Msensor is capable of MEASURING TORQUE
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_MOTION>
    <Name xml:lang="en">MEASURE_MOTION</Name>
    <Definition xml:lang="en">
      An Msensor is capable of MEASURING MOTION
    </Definition>
</Term>

<Term termId="SENSOR_INTELLIGENT_CAMERA>
    <Name xml:lang="en">INTELLIGENT_CAMERA</Name>
    <Definition xml:lang="en">
      An Msensor is capable of being an intelligent camera
    </Definition>
</Term>

<Term termId="SENSOR_DETECT_MULTI_INTERACTION_POINT>
    <Name xml:lang="en">DETECT_MULTI_INTERACTION_POINT</Name>
    <Definition xml:lang="en">
      An Msensor is capable of detecting multi interaction points of hands
    </Definition>
</Term>

<Term termId="SENSOR_DETECT_GAZE>
    <Name xml:lang="en">DETECT_GAZE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of detecting gaze
    </Definition>
</Term>

<Term termId="SENSOR_DETECT_GAZE>
    <Name xml:lang="en">DETECT_GAZE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of detecting gaze
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_WIND_INTENSITY>
    <Name xml:lang="en">MEASURE_WIND_INTENSITY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring wind intensity
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_WIND_DIRECTION>
    <Name xml:lang="en">MEASURE_WIND_DIRECTION</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring wind direction
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ALTITUDE>
    <Name xml:lang="en">MEASURE_ALTITUDE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring altitude
    </Definition>
</Term>
```

```
<Term termId="SENSOR_MEASURE_GAS>
    <Name xml:lang="en">MEASURE_GAS</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring gas
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_DUST>
    <Name xml:lang="en">MEASURE_DUST</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring dust
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BEND>
    <Name xml:lang="en">MEASURE_BEND</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring a degree of bending
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BODY_HEIGHT>
    <Name xml:lang="en">MEASURE_BODY_HEIGHT</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring a body height
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BODY_WEIGHT>
    <Name xml:lang="en">MEASURE_BODY_WEIGHT</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring a body weight
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BODY_TEMPERATURE>
    <Name xml:lang="en">MEASURE_BODY_TEMERATURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring a body temperature
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BODY_FAT>
    <Name xml:lang="en">MEASURE_BODY_FAT</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring a body FAT
    </Definition>
</Term>

<Term termId="SENSOR_CLASSIFY_BLOOD_TYPE>
    <Name xml:lang="en">CLASSIFY_BLOOD_TYPE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of classifying the blood type
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BLOOD_PRESSURE>
    <Name xml:lang="en">MEASURE_BLOOD_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the blood pressure
```

```
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BLOOD_SUGAR>
    <Name xml:lang="en">MEASURE_BLOOD_SUGAR</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the blood sugar
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BLOOD_OXYGEN>
    <Name xml:lang="en">MEASURE_BLOOD_OXYGEN</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the blood oxygen
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_HEARTBIT_RATE>
    <Name xml:lang="en">MEASURE_HEARTBIT_RATE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the heartbit rate
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ELECTROGRAPH>
    <Name xml:lang="en">MEASURE_ELECTROGRAPH</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the electrograph
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_EEG>
    <Name xml:lang="en">MEASURE_EEG</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the EEG
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ECG>
    <Name xml:lang="en">MEASURE_ECG</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the ECG
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_EMG>
    <Name xml:lang="en">MEASURE_EMG</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the EMG
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_EOG>
    <Name xml:lang="en">MEASURE_EOG</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the EOG
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_GSR>
    <Name xml:lang="en">MEASURE_GSR</Name>
```

```
    <Definition xml:lang="en">
      An Msensor is capable of measuring the GSR
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_BIOSIGNAL>
    <Name xml:lang="en">MEASURE_BIOSIGNAL</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the biosignals
    </Definition>
</Term>

<Term termId="SENSOR_DESCRIBE_WEATHER>
    <Name xml:lang="en">DESCRIBE_WEATHER</Name>
    <Definition xml:lang="en">
      An Msensor is capable of describing the weather
    </Definition>
</Term>

<Term termId="SENSOR_DETECT_FACIAL_EXPRESSION>
    <Name xml:lang="en">DETECT_FACIAL_EXPRESSION</Name>
    <Definition xml:lang="en">
      An Msensor is capable of detecting the facial expression
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_FACIAL_MORPHOLOGY>
    <Name xml:lang="en">MEASURE_FACIAL_MORPHOLOGY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the facial morphology
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_FACIAL_EXPRESSION_CHARACTERISTIC>
    <Name xml:lang="en">MEASURE_FACIAL_EXPRESSION_CHARACTERISTIC</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the facial expression characteristic
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_GEOMAGNETIC>
    <Name xml:lang="en">MEASURE_GEOMAGNETIC</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the geomagnetic
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_PROXIMITY>
    <Name xml:lang="en">MEASURE_PROXIMITY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the proximity
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_PROXIMITY>
    <Name xml:lang="en">MEASURE_PROXIMITY</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the proximity
    </Definition>
</Term>
```

```
<Term termId="SENSOR_SWITCH>
    <Name xml:lang="en">SWITCH</Name>
    <Definition xml:lang="en">
      An Msensor is capable of switching
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_SPECTRUM_IMAGE>
    <Name xml:lang="en">CAPTURE_SPECTRUM_IMAGE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the spectrum image
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_SPECTRUM_VIDEO>
    <Name xml:lang="en">CAPTURE_SPECTRUM_VIDEO</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the spectrum video
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_SPECTRUM_VIDEO>
    <Name xml:lang="en">CAPTURE_SPECTRUM_VIDEO</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the spectrum video
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_DEPTH_IMAGE>
    <Name xml:lang="en">CAPTURE_DEPTH_IMAGE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the depth image
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_DEPTH_VIDEO>
    <Name xml:lang="en">CAPTURE_DEPTH_VIDEO</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the depth video
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_STEREO_IMAGE>
    <Name xml:lang="en">CAPTURE_STEREO_IMAGE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the stereo image
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_THERMOGRAPHIC_IMAGE>
    <Name xml:lang="en">CAPTURE_THERMOGRAPHIC_IMAGE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the thermographic image
    </Definition>
</Term>

<Term termId="SENSOR_CAPTURE_THERMOGRAPHIC_VIDEO>
    <Name xml:lang="en">CAPTURE_THERMOGRAPHIC_VIDEO</Name>
    <Definition xml:lang="en">
      An Msensor is capable of capturing the thermographic video
```

```
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ENGINE_OIL_TEMPERATURE>
    <Name xml:lang="en">MEASURE_ENGINE_OIL_TEMPERATURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the engine oil temperature
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_INTAKE_AIR_TEMPERATURE>
    <Name xml:lang="en">MEASURE_INTAKE_AIR_TEMPERATURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the intake air temperature
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_TIRE_PRESSURE>
    <Name xml:lang="en">MEASURE_TIRE_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the tire pressure
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_TIRE_PRESSURE>
    <Name xml:lang="en">MEASURE_TIRE_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the tire pressure
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_DISTANCE_TRAVELLED>
    <Name xml:lang="en">MEASURE_TIRE_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the distance travelled
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_DISTANCE_TRAVELLED>
    <Name xml:lang="en">MEASURE_TIRE_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the distance travelled
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_SPEED>
    <Name xml:lang="en">MEASURE_SPEED</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the speed
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_VEHICLE_SPEED>
    <Name xml:lang="en">MEASURE_VEHICLE_SPEED</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the vehicle speed
    </Definition>
</Term>
```

```
<Term termId="SENSOR_MEASURE_MASS_AIRFLOW>
    <Name xml:lang="en">MEASURE_MASS_AIRFLOW</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the airflow mass
    </Definition>
</Term>

<Term termId="SENSOR_DESCRIBE_PERCENTAGE>
    <Name xml:lang="en">DESCRIBE_PERCENTAGE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of describing in the percentage
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_FUEL_LEVEL>
    <Name xml:lang="en">MEASURE_FUEL_LEVEL</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the fuel level
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_MANIFOLD_ABSOLUTE_PRESSURE>
    <Name xml:lang="en">MEASURE_MANIFOLD_ABSOLUTE_PRESSURE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the manifold absolute pressure
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ENGINE_RPM>
    <Name xml:lang="en">MEASURE_ENGINE_RPM</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the engine RPM
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_CENTER_OF_MASS>
    <Name xml:lang="en">MEASURE_CENTER_OF_MASS</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the center of mass
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_RADAR>
    <Name xml:lang="en">MEASURE_RADAR</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the RADAR
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ARRAY_IMAGE>
    <Name xml:lang="en">MEASURE_ARRAY_IMAGE</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the arrayed image
    </Definition>
</Term>

<Term termId="SENSOR_MEASURE_ARRAY_VIDEO>
    <Name xml:lang="en">MEASURE_ARRAY_VIDEO</Name>
    <Definition xml:lang="en">
      An Msensor is capable of measuring the arrayed video
```

```
    </Definition>
</Term>

<Term termId="SENSOR_DETECT_SCENT>
    <Name xml:lang="en">DETECT_SCENT</Name>
    <Definition xml:lang="en">
      An Msensor is capable of detecting the scent
    </Definition>
</Term>
```

Table A.2 — Binary representation of `SensorCapabilityCS`

Binary representation	Term ID of SensorCapabilityCS
00000000	SENSOR_DEFAULT
00000001	SENSOR_CAPTURE_VIDEO
00000010	SENSOR_CAPTURE_AUDIO
00000011	SENSOR_STREAM_VIDEO
00000100	SENSOR_STREAM_STEREO_VIDEO
00000101	SENSOR_STREAM_AUDIO
00000110	SENSOR_MEASURE_DISTANCE
00000111	SENSOR_MEASURE_LATITUDE
00001000	SENSOR_MEASURE_LONGITUDE
00001001	SENSOR_MEASURE_AZIMUTH
00001010	SENSOR_MEASURE_YAW_ANGLE
00001011	SENSOR_MEASURE_PITCH_ANGLE
00001100	SENSOR_MEASURE_ROLL_ANGLE
00001101	SENSOR_CAPTURE_STEREO_VIDEO
00001110	SENSOR_CAPTURE_IMAGE
00001111	SENSOR_MEASURE_GLOBAL_POSITION
00010000	SENSOR_READ_RFID
00010001	SENSOR_DETECT_POSITION
00010010	SENSOR_MEASURE_LIGHT_INTENSITY
00010011	SENSOR_MEASURE_LIGHT_COLOUR
00010100	SENSOR_MEASURE_AMBIENT_NOISE

Binary representation	Term ID of SensorCapabilityCS
00010101	SENSOR_MEASURE_TEMPERATURE
00010110	SENSOR_MEASURE_HUMIDITY
00010111	SENSOR_MEASURE_ATMOSPHERIC_PRESSURE
00011000	SENSOR_MEASURE_PRESSURE
00011001	SENSOR_MEASURE_VELOCITY
00011010	SENSOR_MEASURE_ACCELERATION
00011011	SENSOR_MEASURE_AGULAR_ACCELERATION
00011100	SENSOR_MEASURE_AGULAR_VELOCITY
00011101	SENSOR_MEASURE_FORCE
00011110	SENSOR_MEASURE_TORQUE
00011111	SENSOR_MEASURE_MOTION
00100000	SENSOR_INTELLIGENT_CAMERA
00100001	SENSOR_DETECT_MULTI_INTERACTION_POINT
00100010	SENSOR_DETECT_GAZE
00100011	SENSOR_MEASURE_WIND_INTENSITY
00100100	SENSOR_MEASURE_WIND_DIRECTION
00100101	SENSOR_MEASURE_ALTITUDE
00100110	SENSOR_MEASURE_GAS
00100111	SENSOR_MEASURE_DUST
00101000	SENSOR_MEASURE_BEND
00101001	SENSOR_MEASURE_BODY_HEIGHT
00101010	SENSOR_MEASURE_BODY_WEIGHT
00101011	SENSOR_MEASURE_BODY_TEMPERATURE
00101100	SENSOR_MEASURE_BODY_FAT
00101101	SENSOR_CLASSIFY_BLOOD_TYPE
00101110	SENSOR_MEASURE_BLOOD_PRESSURE
00101111	SENSOR_MEASURE_BLOOD_SUGAR

Binary representation	Term ID of SensorCapabilityCS
00110000	SENSOR_MEASURE_BLOOD_OXYGEN
00110001	SENSOR_MEASURE_HEARTBIT_RATE
00110010	SENSOR_MEASURE_ELECTROGRAPH
00110011	SENSOR_MEASURE_EEG
00110100	SENSOR_MEASURE_ECG
00110101	SENSOR_MEASURE_EMG
00110110	SENSOR_MEASURE_EOG
00110111	SENSOR_MEASURE_GSR
00111000	SENSOR_MEASURE_BIOSIGNAL
00111001	SENSOR_DESCRIBE_WEATHER
00111010	SENSOR_DETECT_FACIAL_EXPRESSION
00111011	SENSOR_MEASURE_FACIAL_MORPHOLOGY
00111100	SENSOR_MEASURE_FACIAL_EXPRESSION_CHARACTERISTIC
00111101	SENSOR_MEASURE_GEOMAGNETIC
00111110	SENSOR_MEASURE_PROXIMITY
00111111	SENSOR_SWITCH
01000000	SENSOR_CAPTURE_SPECTRUM_IMAGE
01000001	SENSOR_CAPTURE_SPECTRUM_VIDEO
01000010	SENSOR_CAPTURE_DEPTH_IMAGE
01000011	SENSOR_CAPTURE_DEPTH_VIDEO
01000100	SENSOR_CAPTURE_STEREO_IMAGE
01000101	SENSOR_CAPTURE_THERMOGRPHIC_IMAGE
01000110	SENSOR_CAPTURE_THERMOGRPHIC_VIDEO
01000111	SENSOR_MEASURE_ENGINE_OIL_TEMPERATURE
01001000	SENSOR_MEASURE_INTAKE_AIR_TEMPERATURE
01001001	SENSOR_MEASURE_TIRE_PRESSURE
01001010	SENSOR_MEASURE_DISTANCE_TRAVELLED

Binary representation	Term ID of SensorCapabilityCS
01001011	SENSOR_MEASURE_SPEED
01001100	SENSOR_MEASURE_VEHICLE_SPEED
01001101	SENSOR_MEASURE_MASS_AIRFLOW
01001110	SENSOR_DESCRIBE_PERCENTAGE
01001111	SENSOR_MEASURE_FUEL_LEVEL
01010000	SENSOR_MEASURE_MANIFOLD_ABSOLUTE_PRESSURE
01010001	SENSOR_MEASURE_ENGINE_RPM
01010010	SENSOR_MEASURE_CENTER_OF_MASS
01010011	SENSOR_MEASURE_RADAR
01010100	SENSOR_CAPTURE_ARRAY_IMAGE
01010101	SENSOR_CAPTURE_ARRAY_VIDEO
01010110	SENSOR_DETECT_SCENT
01010111 - 11111111	Reserved

A.2.2 ActuatorCapabilityCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-ActuatorCapabilityCS-NS">

<Term termId="ACTUATOR_DEFAULT">
    <Name xml:lang="en">DEFAULT</Name>
    <Definition xml:lang="en">
       Any MActuators
    </Definition>
</Term>

<Term termId="ACTUATOR_VIBRATE">
    <Name xml:lang="en">VIBRATE</Name>
    <Definition xml:lang="en">
       An MActuator is capable of vibrating
    </Definition>
</Term>

<Term termId="ACTUATOR_PLAY_AUDIO">
    <Name xml:lang="en">PLAY_AUDIO</Name>
    <Definition xml:lang="en">
       An MActuator is capable of play an audio
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_COLOR_TEMPERATURE">
    <Name xml:lang="en">CHANGE_COLOR_TEMPERATURE</Name>
    <Definition xml:lang="en">
```

```
        An MActuator is capable of changing the color temperature
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_SATURATION">
    <Name xml:lang="en">CHANGE_SATURATION</Name>
    <Definition xml:lang="en">
        An MActuator is capable of changing the saturation
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_BRIGHTNESS">
    <Name xml:lang="en">CHANGE_BRIGHTNESS</Name>
    <Definition xml:lang="en">
        An MActuator is capable of changing the brightness
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_HUE">
    <Name xml:lang="en">CHANGE_HUE</Name>
    <Definition xml:lang="en">
        An MActuator is capable of changing the hue
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_VOLUME">
    <Name xml:lang="en">CHANGE_VOLUME</Name>
    <Definition xml:lang="en">
        An MActuator is capable of changing the volume
    </Definition>
</Term>

<Term termId="ACTUATOR_PLAY_VIDEO">
    <Name xml:lang="en">PLAY_VIDEO</Name>
    <Definition xml:lang="en">
        An MActuator is capable of playing a video
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_ZOOM">
    <Name xml:lang="en">CHANGE_ZOOM</Name>
    <Definition xml:lang="en">
        An MActuator is capable of zooming
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_RESOLUTION">
    <Name xml:lang="en">CHANGE_RESOLUTION</Name>
    <Definition xml:lang="en">
        An MActuator is capable of changing resolutions
    </Definition>
</Term>

<Term termId="ACTUATOR_CHANGE_ORIENTATION">
<Name xml:lang="en">CHANGE_ORIENTATION</Name>
    <Definition xml:lang="en">
        An MActuator is capable of changing the orientation
    </Definition>
</Term>
```

```
<Term termId="ACTUATOR_REWIND">
    <Name xml:lang="en">REWIND</Name>
    <Definition xml:lang="en">
       An MActuator is capable of rewinding a video or an audio
    </Definition>
</Term>

<Term termId="ACTUATOR_FASTFORWARD">
    <Name xml:lang="en">FASTFORWARD</Name>
    <Definition xml:lang="en">
       An MActuator is capable of fastforwarding a video or an audio
    </Definition>
</Term>
<Term termId="ACTUATOR_LIGHTING">
    <Name xml:lang="en">LIGHTING</Name>
    <Definition xml:lang="en">
       An MActuator is capable of turning on/off a light
    </Definition>
</Term>

<Term termId="ACTUATOR_ARRAYED_LIGHTING">
    <Name xml:lang="en">ARRAYED_LIGHTING</Name>
    <Definition xml:lang="en">
       An MActuator is capable of turning on/off an arrayed light
    </Definition>
</Term>

<Term termId="ACTUATOR_SPRAY_WATER">
    <Name xml:lang="en">SPRAY_WATER</Name>
    <Definition xml:lang="en">
       An MActuator is capable of spraying water
    </Definition>
</Term>

<Term termId="ACTUATOR_SPRAY_SCENT">
    <Name xml:lang="en">SPRAY_SCENT</Name>
    <Definition xml:lang="en">
       An MActuator is capable of spraying scent
    </Definition>
</Term>

<Term termId="ACTUATOR_SPRAY_FOG">
    <Name xml:lang="en">SPRAY_FOG</Name>
    <Definition xml:lang="en">
       An MActuator is capable of spraying fog
    </Definition>
</Term>

<Term termId="ACTUATOR_SPRAY_BUBBLE">
    <Name xml:lang="en">SPRAY_BUBBLE</Name>
    <Definition xml:lang="en">
       An MActuator is capable of spraying bubble
    </Definition>
</Term>

<Term termId="ACTUATOR_DISPLAY_TRANSPARENT">
    <Name xml:lang="en">DISPLAY_TRANSPARENT</Name>
    <Definition xml:lang="en">
       A transparent display which can see-through
    </Definition>
</Term>
```

```
<Term termId="ACTUATOR_DISPLAY_OPAQUE">
    <Name xml:lang="en">DISPLAY_OPAQUE</Name>
    <Definition xml:lang="en">
       An opaque display which can't support the see-through
   </Definition>
</Term>
```

Table A.3 — Binary representation of `ActuatorCapabilityCS`

Binary representation	Term ID of ActuatorCapabilityCS
00000000	ACTUATOR_DEFAULT
00000001	ACTUATOR_VIBRATE
00000010	ACTUATOR_PLAY_AUDIO
00000011	ACTUATOR_CHANGE_COLOR_TEMPERATURE
00000100	ACTUATOR_CHANGE_SATURATION
00000101	ACTUATOR_CHANGE_BRIGHTNESS
00000110	ACTUATOR_CHANGE_HUE
00000111	ACTUATOR_CHANGE_VOLUME
00001000	ACTUATOR_PLAY_VIDEO
00001001	ACTUATOR_CHANGE_ZOOM
00001010	ACTUATOR_CHANGE_RESOLUTION
00001011	ACTUATOR_CHANGE_ORIENTATION
00001100	ACTUATOR_REWIND
00001101	ACTUATOR_FASTFORWARD
00001110	ACTUATOR_LIGHTING
00001111	ACTUATOR_ARRAYED_LIGHTING
00010000	ACTUATOR_SPRAY_WATER
00010001	ACTUATOR_SPRAY_SCENT
00010010	ACTUATOR_SPARY_FOG
00010011	ACTUATOR_SPRAY_BUBBLE
00010100	ACTUATOR _DISPLAY_TRANSPARENT
00010101	ACTUATOR _DISPLAY_OPAQUE
	ACTUATOR _HAND_GESTURE_COMMAND

Binary representation	Term ID of ActuatorCapabilityCS
00010110	ACTUATOR_HEATING
00010111	ACTUATOR_COOLING
00011000	ACTUATOR_WIND
00011001	ACTUATOR_MOTION_CHAIR
00011010	ACTUATOR_TACTILE_EFFECT
00011011	ACTUATOR_3D_PRINT
00011100 - 11111111	Reserved

A.2.3 AnalyserCapabilityCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-AnalyserCapabilityCS-NS">

<Term termId="ANALYSER_DEFAULT">
    <Name xml:lang="en">DEFAULT</Name>
    <Definition xml:lang="en">
       Any MAnalysers
    </Definition>
</Term>

<Term termId="ANALYSER_SYNCRONISE_TIME">
    <Name xml:lang="en">SYNCRONISE_TIME</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of sychroising time with multiple videos
    </Definition>
</Term>

<Term termId="ANALYSER_DETECT_SOCIAL_EVENT">
    <Name xml:lang="en">DETECT_SOCIAL_EVENT</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of detecting a social event in a video
    </Definition>
</Term>

<Term termId="ANALYSER_DETECT_HAND_GESTURE">
    <Name xml:lang="en">DETECT_HAND_GESTURE</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of detecting hand gestures
    </Definition>
</Term>

<Term termId="ANALYSER_RECOGNISE_HAND_GESTURE">
    <Name xml:lang="en">RECOGNISE_HAND_GESTURE</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of recognizing hand gestures
    </Definition>
</Term>
```

```
<Term termId="ANALYSER_MAP_HAND_GESTURE_COMMAND">
    <Name xml:lang="en">MAP_HAND_GESTURE_COMMAND</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of mapping a hand gesture to the corresponding
command
    </Definition>
</Term>

<Term termId="ANALYSER_GENERATE_HEALTHCARE_INFO">
    <Name xml:lang="en">GENERATE_HEALTHCARE_INFO</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of generating healthcare information
    </Definition>
</Term>

<Term termId="ANALYSER_RECOGNISE_SPEECH">
    <Name xml:lang="en">RECOGNISE_SPEECH</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of recognizing a speech
    </Definition>
</Term>

<Term termId="ANALYSER_ANALYSE_QUESTION">
    <Name xml:lang="en">ANALYSE_QUESTION</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of analyse questions
    </Definition>
</Term>

<Term termId="ANALYSER_RECOGNISE_SCENT_IMAGE">
    <Name xml:lang="en">RECOGNISE_SCENT_IMAGE</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of recognizing scent objects or scenes from an
image
    </Definition>
</Term>

<Term termId="ANALYSER_CONVERT_TEXT_TO_SPEECH">
    <Name xml:lang="en">CONVERT_TEXT_TO_SPEECH</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of converting text to speech
    </Definition>
</Term>

<Term termId="ANALYSER_DETECT_COLLISION">
    <Name xml:lang="en">DETECT_COLLISION</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of detecting collisions
    </Definition>
</Term>

<Term termId="ANALYSER_GUIDE_DIRECTION">
    <Name xml:lang="en">GUIDE_DIRECTION</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of guiding directions
    </Definition>
</Term>
```

```
<Term termId="ANALYSER_COUNTING_PEOPLE">
    <Name xml:lang="en">COUNTING_PEOPLE</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of counting people in the video
    </Definition>
</Term>

<Term termId="ANALYSER_CALCULATE_MUSIC_FREQUENCY">
    <Name xml:lang="en">CALCULATE_MUSIC_FREQUENCY</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of calculating music frequency
    </Definition>
</Term>

<Term termId="ANALYSER_CALCULATE_LIGHT_COLOUR_INFO">
    <Name xml:lang="en">CALCULATE_LIGHT_COLOUR_INFO</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of calculating light colour
    </Definition>
</Term>

<Term termId="ANALYSER_GENERATE_VIDEO_CONTENT_CLASS">
    <Name xml:lang="en">GENERATE_VIDEO_CONTENT_CLASS</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of generating a video contrent class from a video
    </Definition>
</Term>

<Term termId="ANALYSER_DETECT_FACE">
    <Name xml:lang="en">DETECT_FACE</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of detecting faces from a video stream
    </Definition>
</Term>

<Term termId="ANALYSER_VERIFY_FACE">
    <Name xml:lang="en">VERIFY_FACE</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of verifying a face from a video stream
    </Definition>
</Term>

<Term termId="ANALYSER_GENERATE_SECURITY_ALERT">
    <Name xml:lang="en">GENERATE_SECURITY_ALERT</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of generating security alerts
    </Definition>
</Term>

<Term termId="ANALYSER_GENERATE_SECURITY_TITLE">
    <Name xml:lang="en">GENERATE_SECURITY_TITLE</Name>
    <Definition xml:lang="en">
       An MAnalyser is capable of generating security titles
    </Definition>
</Term>
```

Table A.4 — Binary representation of `AnalyserCapabilityCS`

Binary representation	Term ID of AnalyserCapabilityCS
00000000	ANALYSER_DEFAULT
00000001	ANALYSER_SYCHRONISE_TIME
00000010	ANALYSER_DETECT_SOCIAL_EVENT
00000011	ANALYSER_DETECT_HAND_GESTURE
00000100	ANALYSER_RECOGNISE_HAND_GESTURE
00000101	ANALYSER_MAP_HAND_GESTURE_COMMAND
00000110	ANALYSER_GENERATE_HEALTHCARE_INFO
00000111	ANALYSER_RECOGNISE_SPEECH
00001000	ANALYSER_ANALYSE_QUESTION
00001001	ANALYSER_RECOGNISE_SCENT_IMAGE
00001010	ANALYSER_CONVERT_TEXT_TO_SPEECH
00001011	ANALYSER_DETECT_COLLISION
00001100	ANALYSER_GUIDE_DIRECTION
00001101	ANALYSER_COUNTING_PEOPLE
00001110	ANALYSER_CALCULATE_MUSIC_FREQUENCY
00001111	ANALYSER_CALCULATE_LIGHT_COLOUR_INFO
00010000	ANALYSER_GENERATE_VIDEO_CONTENT_CLASS
00010001	ANALYSER_DETECT_FACE
00010010	ANALYSER_VERIFY_FACE
00010011	ANALYSER_GENERATE_SECURITY_ALERT
00010100	ANALYSER_GENERATE_SECURITY_TITLE
00010101 - 11111111	Reserved

A.2.4 StorageCapabilityCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-StorageCapabilityCS-NS">

<Term termId="STORAGE_DEFAULT">
    <Name xml:lang="en">STORAGE_DEFAULT</Name>
    <Definition xml:lang="en">
       Any MStorages
    </Definition>
</Term>

<Term termId="STORAGE_SAVE">
    <Name xml:lang="en">STORAGE_SAVE</Name>
    <Definition xml:lang="en">
       An MStorage is capable of saving data
    </Definition>
</Term>

<Term termId="STORAGE_READ">
    <Name xml:lang="en">STORAGE_READ</Name>
    <Definition xml:lang="en">
       An MStorage is capable of reading data
    </Definition>
</Term>

<Term termId="STORAGE_DELETE">
    <Name xml:lang="en">STORAGE_DELETE</Name>
    <Definition xml:lang="en">
       An MStorage is capable of deleting data
    </Definition>
</Term>

<Term termId="STORAGE_UPDATE">
    <Name xml:lang="en">STORAGE_UPDATE</Name>
    <Definition xml:lang="en">
       An MStorage is capable of updating data
    </Definition>
</Term>
```


Table A.5 — Binary representation of `StorageCapabilityCS`

Binary representation	Term ID of StorageCapabilityCS
00000000	STORAGE_DEFAULT
00000001	STORAGE_SAVE
00000010	STORAGE_READ
00000011	STORAGE_DELETE
00000100	STORAGE_UPDATE
00000101 - 11111111	Reserved

A.2.5 ManagerCapabilityCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-ManagerCapabilityCS-NS">

<Term termId="MANAGER_DEFAULT">
    <Name xml:lang="en">DEFAULT</Name>
    <Definition xml:lang="en">
       Any MManager
    </Definition>
</Term>

<Term termId="MANAGER_RETRIEVE_MTHINGS_BY_CAPABILITY">
    <Name xml:lang="en">RETRIEVE_MTHINGS_BY_CAPABILITY</Name>
    <Definition xml:lang="en">
       An MManager is capable of searching MThings by capability
    </Definition>
</Term>

<Term termId="MANAGER_RETRIEVE_MTHINGS_BY_TOKEN">
    <Name xml:lang="en">RETRIEVE_MTHINGS_BY_TOKEN</Name>
    <Definition xml:lang="en">
       An MManager is capable of searching MThings by token
    </Definition>
</Term>

<Term termId="MANAGER_REGISTER_MTHING">
    <Name xml:lang="en">REGISTER_MTHING</Name>
    <Definition xml:lang="en">
       An MManager is capable of registering an MThing
    </Definition>
</Term>

<Term termId="MANAGER_UNREGISTER_MTHING">
    <Name xml:lang="en">UNREGISTER_MTHING</Name>
    <Definition xml:lang="en">
       An MManager is capable of unregistering an MThing
    </Definition>
</Term>

<Term termId="MANAGER_SHOW_REGISTERED_MTHINGS">
    <Name xml:lang="en">SHOW_REGISTERED_MTHINGS</Name>
    <Definition xml:lang="en">
       An MManager is capable of showing registered MThings
    </Definition>
</Term>
```


Table A.6 — Binary representation of `ManagerCapabilityCS`

Binary representation	Term ID of ManagerCapabilityCS
00000000	MANAGER_DEFAULT
00000001	MANAGER_RETRIEVE_MTHINGS_BY_CAPABILITY
00000010	MANAGER_RETRIEVE_MTHINGS_BY_TOKEN
00000011	MANAGER_REGISTER_MTHING

Binary representation	Term ID of ManagerCapabilityCS
00000100	MANAGER_UNREGISTER_MTHING
00000101	MANAGER_SHOW_REGISTERED_MTHINGS
00000110 - 11111111	Reserved

A.2.6 AggregatorCapabilityCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-AggregatorCapabilityCS-NS">

<Term termId="AGGREGATOR_DEFAULT">
    <Name xml:lang="en">AGGREGATOR_DEFAULT</Name>
    <Definition xml:lang="en">
       Any MAggregator
    </Definition>
</Term>

<Term termId="AGGREGATOR_INCLUDE_MTHING">
    <Name xml:lang="en">AGGREGATOR_INCLUDE_MTHING</Name>
    <Definition xml:lang="en">
       An MAggregator is capable of including an MThing
    </Definition>
</Term>

<Term termId="AGGREGATOR_EXCLUDE_MTHING">
    <Name xml:lang="en">AGGREGATOR_EXCLUDE_MTHING</Name>
    <Definition xml:lang="en">
       An MAggregator is capable of excluding an MThing
    </Definition>
</Term>

<Term termId="AGGREGATOR_SHOW_AGGREGATED_MTHINGS">
    <Name xml:lang="en">AGGREGATOR_SHOW_AGGREGATED_MTHINGS</Name>
    <Definition xml:lang="en">
       An MAggregator is capable of showing a list of MThings aggregated
    </Definition>
</Term>
```


Table A.7 — Binary representation of `AggregatorCapabilityCS`

Binary representation	Term ID of AggregatorCapabilityCS
00000000	AGGREGATOR_DEFAULT
00000001	AGGREGATOR_INCLUDE_MTHING
00000010	AGGREGATOR_EXCLUDE_MTHING
00000011	AGGREGATOR_SHOW_AGGREGATED_MTHINGS
00000100 - 11111111	Reserved

A.3 CapabilityParameterCS

A.3.1 SensorCapabilityParameterCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-SensorCapabilityParameterCS-NS">

<Term termId="AUDIO_CODEC_PCM">
    <Name xml:lang="en">PCM Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in PCM codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_AC3">
    <Name xml:lang="en">AC3 Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in AC3 codec
    </Definition>
</Term>

<Term termId="AUDIO_STREAMING_PROTOCOL_HTTP">
    <Name xml:lang="en">Audio Streaming HTTP Protocol</Name>
    <Definition xml:lang="en">
       Audio is streamed in HTTP protocol
    </Definition>
</Term>

<Term termId="AUDIO_STREAMING_PROTOCOL_RTP">
    <Name xml:lang="en">Audio Streaming RTP Protocol</Name>
    <Definition xml:lang="en">
       Audio is streamed in RTP Protocol
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_MP3">
    <Name xml:lang="en">MP3 Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in MP3 codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_AAC">
    <Name xml:lang="en">AAC Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in AAC codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_APE">
    <Name xml:lang="en">APE Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in APE codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_WMA">
    <Name xml:lang="en">WMA Codec</Name>
    <Definition xml:lang="en">
```

```
        Audio is streamed in WMA codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_MPC">
    <Name xml:lang="en">MPC Codec</Name>
    <Definition xml:lang="en">
        Audio is streamed in MPC codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_FLAC">
    <Name xml:lang="en">FLAC Codec</Name>
    <Definition xml:lang="en">
        Audio is streamed in FLAC codec
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_MP4_AVC">
    <Name xml:lang="en">mpeg-4 AVC Codec</Name>
    <Definition xml:lang="en">
        Video is streamed in mpeg-4 AVC codec
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_MPEG1">
    <Name xml:lang="en">MPEG1 Codec</Name>
    <Definition xml:lang="en">
        Video is streamed in MPEG1 codec
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_MPEG2">
    <Name xml:lang="en">MPEG2 Codec</Name>
    <Definition xml:lang="en">
        Video is streamed in MPEG2 codec
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_MPEG4_ASP">
    <Name xml:lang="en">MPEG4 ASP Codec</Name>
    <Definition xml:lang="en">
        Video is streamed in MPEG4 ASP codec
    </Definition>
</Term>

<Term termId="VIDEO_STREAMING_PROTOCOL_HTTP">
    <Name xml:lang="en">VIDEO Streaming HTTP Protocol</Name>
    <Definition xml:lang="en">
        VIDEO is streamed in HTTP protocol.
    </Definition>
</Term>

<Term termId="VIDEO_STREAMING_PROTOCOL_RTP">
    <Name xml:lang="en">VIDEO Streaming RTP Protocol</Name>
    <Definition xml:lang="en">
        VIDEO is streamed in RTP Protocol.
    </Definition>
</Term>
```

Table A.8 — Binary representation of `SensorCapabilityParameterCS`

Binary representation	Term ID of SensorCapabilityParameterCS
00000000	AUDIO_CODEC_PCM
00000001	AUDIO_CODEC_AC3
00000010	AUDIO_STREAMING_PROTOCOL_HTTP
00000011	AUDIO_STREAMING_PROTOCOL_RTP
00000100	AUDIO_CODEC_MP3
00000101	AUDIO_CODEC_AAC
00000110	AUDIO_CODEC_APE
00000111	AUDIO_CODEC_WMA
00001000	AUDIO_CODEC_MPC
00001001	AUDIO_CODEC_FLAC
00001010	VIDEO_CODEC_MP4_AVC
00001011	VIDEO_CODEC_MPEG1
00001100	VIDEO_CODEC_MPEG2
00001101	VIDEO_CODEC_MPEG4_ASP
00001110	VIDEO_STREAMING_PROTOCOL_HTTP
00001111	VIDEO_STREAMING_PROTOCOL_RTP
00010000 - 11111111	Reserved

A.3.2 ActuatorCapabilityParameterCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-ActuatorCapabilityParameterCS-
NS">

<Term termId="AUDIO_CODEC_PCM">
   <Name xml:lang="en">PCM Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in PCM codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_AC3">
    <Name xml:lang="en">AC3 Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in AC3 codec
    </Definition>
</Term>
```

```
<Term termId="AUDIO_STREAMING_PROTOCOL_HTTP">
    <Name xml:lang="en">Audio Streaming HTTP Protocol</Name>
    <Definition xml:lang="en">
       Audio is streamed in HTTP protocol
    </Definition>
</Term>

<Term termId="AUDIO_STREAMING_PROTOCOL_RTP">
    <Name xml:lang="en">Audio Streaming RTP Protocol</Name>
    <Definition xml:lang="en">
       Audio is streamed in RTP Protocol
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_MP3">
    <Name xml:lang="en">MP3 Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in MP3 codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_AAC">
    <Name xml:lang="en">AAC Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in AAC codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_APE">
    <Name xml:lang="en">APE Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in APE codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_WMA">
    <Name xml:lang="en">WMA Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in WMA codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_MPC">
    <Name xml:lang="en">MPC Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in MPC codec
    </Definition>
</Term>

<Term termId="AUDIO_CODEC_FLAC">
    <Name xml:lang="en">FLAC Codec</Name>
    <Definition xml:lang="en">
       Audio is streamed in FLAC codec
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_H264">
    <Name xml:lang="en">H264 Codec</Name>
    <Definition xml:lang="en">
       Video is streamed in H.264 codec
```

```
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_MPEG1">
    <Name xml:lang="en">MPEG1 Codec</Name>
    <Definition xml:lang="en">
       Video is streamed in MPEG1 codec
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_MPEG2">
    <Name xml:lang="en">MPEG2 Codec</Name>
    <Definition xml:lang="en">
       Video is streamed in MPEG2 codec
    </Definition>
</Term>

<Term termId="VIDEO_CODEC_MPEG4_ASP">
    <Name xml:lang="en">MPEG4 ASP Codec</Name>
    <Definition xml:lang="en">
       Video is streamed in MPEG4 ASP codec
    </Definition>
</Term>
```

Table A.9 — Binary representation of `ActuatorCapabilityParameterCS`

Binary representation	Term ID of ActuatorCapabilityParameterCS
00000000	AUDIO_CODEC_PCM
00000001	AUDIO_CODEC_AC3
00000010	AUDIO_STREAMING_PROTOCOL_HTTP
00000011	AUDIO_STREAMING_PROTOCOL_RTP
00000100	AUDIO_CODEC_MP3
00000101	AUDIO_CODEC_AAC
00000110	AUDIO_CODEC_APE
00000111	AUDIO_CODEC_WMA
00001000	AUDIO_CODEC_MPC
00001001	AUDIO_CODEC_FLAC
00001010	VIDEO_CODEC_H264
00001011	VIDEO_CODEC_MPEG1
00001100	VIDEO_CODEC_MPEG2
00001101	VIDEO_CODEC_MPEG4_ASP
00001110 - 11111111	Reserved

A.3.3 AnalyserCapabilityParameterCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-AnalyserCapabilityParameterCS-
NS">

<Term termId="VIDEO">
    <Name xml:lang="en">VIDEO</Name>
    <Definition xml:lang="en">
       Video is an input to be analysed
    </Definition>
</Term>

<Term termId="AUDIO">
    <Name xml:lang="en">AUDIO</Name>
    <Definition xml:lang="en">
       Audio is an input to be analysed
    </Definition>
</Term>

<Term termId="IMAGE">
    <Name xml:lang="en">IMAGE</Name>
    <Definition xml:lang="en">
       Image is an input to be analysed
    </Definition>
</Term>

<Term termId="TEXT">
    <Name xml:lang="en">TEXT</Name>
    <Definition xml:lang="en">
       Text is an input to be analysed
    </Definition>
</Term>

<Term termId="3D_GRAPHICS">
    <Name xml:lang="en">3D_GRAPHICS</Name>
    <Definition xml:lang="en">
       3D graphics are an input to be analysed
    </Definition>
</Term>

<Term termId="SENSED_DATA_EXCEPT_MEDIA">
    <Name xml:lang="en">SENSED_DATA_EXCEPT_MEDIA</Name>
    <Definition xml:lang="en">
       Sensed data except media data are an input to be analysed
    </Definition>
</Term>
```


Table A.10 — Binary representation of `AnalyserCapabilityParameterCS`

Binary representation	Term ID of AnalyserCapabilityParameterCS
00000000	VIDEO
00000001	AUDIO
00000010	IMAGE
00000011	TEXT

Binary representation	Term ID of AnalyserCapabilityParameterCS
00000100	3D_GRAPHICS
00000101	SENSED_DATA_EXCEPT_MEDIA
00000110 - 11111111	Reserved

A.3.4 StorageCapabilityParameterCS

```
<ClassificationScheme url="urn:mpeg:mpeg-IoMT:01-StorageCapabilityParameterCS-
NS">

<Term termId="DOC">
    <Name xml:lang="en">DOC</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Microsoft Word Document
    </Definition>
</Term>

<Term termId="DOCX">
    <Name xml:lang="en">DOCX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Microsoft Word Open XML Document
    </Definition>
</Term>

<Term termId="LOG">
    <Name xml:lang="en">LOG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Log File
    </Definition>
</Term>

<Term termId="MSG">
    <Name xml:lang="en">MSG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Outlook Mail Message
    </Definition>
</Term>

<Term termId="ODT">
    <Name xml:lang="en">ODT</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing OpenDocument Text Document
    </Definition>
</Term>

<Term termId="PAGES">
    <Name xml:lang="en">PAGES</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Pages Document
    </Definition>
</Term>

<Term termId="RTF">
    <Name xml:lang="en">RTF</Name>
    <Definition xml:lang="en">
```

```
      An MStorage is capable of storing Rich Text Format File
    </Definition>
</Term>

<Term termId="TEX">
    <Name xml:lang="en">TEX</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing LaTeX Source Document
    </Definition>
</Term>

<Term termId="TXT">
    <Name xml:lang="en">TXT</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing Plain Text File
    </Definition>
</Term>

<Term termId="WPD">
    <Name xml:lang="en">WPD</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing WordPerfect Document
    </Definition>
</Term>

<Term termId="WPS">
    <Name xml:lang="en">WPS</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing Microsoft Works Word Processor Document
    </Definition>
</Term>

<Term termId="CSV">
    <Name xml:lang="en">CSV</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing Comma Separated Values File
    </Definition>
</Term>

<Term termId="DAT">
    <Name xml:lang="en">DAT</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing Data File
    </Definition>
</Term>

<Term termId="GED">
    <Name xml:lang="en">GED</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing GEDCOM Genealogy Data File
    </Definition>
</Term>

<Term termId="KEY">
    <Name xml:lang="en">KEY</Name>
    <Definition xml:lang="en">
      An MStorage is capable of storing Keynote Presentation
    </Definition>
</Term>
```

```
<Term termId="KEYCHAIN">
    <Name xml:lang="en">KEYCHAIN</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Mac OS X Keychain File
    </Definition>
</Term>

<Term termId="PPS">
    <Name xml:lang="en">PPS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing PowerPoint Slide Show
    </Definition>
</Term>

<Term termId="PPT">
    <Name xml:lang="en">PPT</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing PowerPoint Presentation
    </Definition>
</Term>

<Term termId="PPTX">
    <Name xml:lang="en">PPTX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing PowerPoint Open XML Presentation
    </Definition>
</Term>

<Term termId="SDF">
    <Name xml:lang="en">SDF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Standard Data File
    </Definition>
</Term>

<Term termId="TAR">
    <Name xml:lang="en">TAR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Consolidated Unix File Archive
    </Definition>
</Term>

<Term termId="TAX2016">
    <Name xml:lang="en">TAX2016</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing TurboTax 2016 Tax Return
    </Definition>
</Term>

<Term termId="TAX2017">
    <Name xml:lang="en">TAX2017</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing TurboTax 2017 Tax Return
    </Definition>
</Term>

<Term termId="VCF">
    <Name xml:lang="en">VCF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing vCard File
```

```
    </Definition>
</Term>

<Term termId="XML">
    <Name xml:lang="en">XML</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing XML File
    </Definition>
</Term>

<Term termId="AIF">
    <Name xml:lang="en">AIF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Audio Interchange File Format
    </Definition>
</Term>

<Term termId="IFF">
    <Name xml:lang="en">IFF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Interchange File Format
    </Definition>
</Term>

<Term termId="M3U">
    <Name xml:lang="en">M3U</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Media Playlist File
    </Definition>
</Term>

<Term termId="M4A">
    <Name xml:lang="en">M4A</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing MPEG-4 Audio File
    </Definition>
</Term>

<Term termId="MID">
    <Name xml:lang="en">MID</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing MIDI File
    </Definition>
</Term>

<Term termId="MP3">
    <Name xml:lang="en">MP3</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing MP3 Audio File
    </Definition>
</Term>

<Term termId="MPA">
    <Name xml:lang="en">MPA</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing MPEG-2 Audio File
    </Definition>
</Term>
```

```
<Term termId="WAV">
    <Name xml:lang="en">WAV</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing WAVE Audio File
    </Definition>
</Term>

<Term termId="WMA">
    <Name xml:lang="en">WMA</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Windows Media Audio File
    </Definition>
</Term>

<Term termId="3G2">
    <Name xml:lang="en">3G2</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing 3GPP2 Multimedia File
    </Definition>
</Term>

<Term termId="3GP">
    <Name xml:lang="en">3GP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing 3GPP Multimedia File
    </Definition>
</Term>

<Term termId="ASF">
    <Name xml:lang="en">ASF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Advanced Systems Format File
    </Definition>
</Term>

<Term termId="AVI">
    <Name xml:lang="en">AVI</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Audio Video Interleave File
    </Definition>
</Term>

<Term termId="FLV">
    <Name xml:lang="en">FLV</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Animate Video File
    </Definition>
</Term>

<Term termId="M4V">
    <Name xml:lang="en">M4V</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing iTunes Video File
    </Definition>
</Term>

<Term termId="MOV">
    <Name xml:lang="en">MOV</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Apple QuickTime Movie
```

```
    </Definition>
</Term>

<Term termId="MP4">
    <Name xml:lang="en">MP4</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing MPEG-4 Video File
    </Definition>
</Term>

<Term termId="MPG">
    <Name xml:lang="en">MPG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing MPEG Video File
    </Definition>
</Term>

<Term termId="RM">
    <Name xml:lang="en">RM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing RealMedia File
    </Definition>
</Term>

<Term termId="SRT">
    <Name xml:lang="en">SRT</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing SubRip Subtitle File
    </Definition>
</Term>

<Term termId="SWF">
    <Name xml:lang="en">SWF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Shockwave Flash Movie
    </Definition>
</Term>

<Term termId="VOB">
    <Name xml:lang="en">VOB</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing DVD Video Object File
    </Definition>
</Term>

<Term termId="WMV">
    <Name xml:lang="en">WMV</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Windows Media Video File
    </Definition>
</Term>

<Term termId="3DM">
    <Name xml:lang="en">3DM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Rhino 3D Model
    </Definition>
</Term>
```

```
<Term termId="3DS">
    <Name xml:lang="en">3DS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing 3D Studio Scene
    </Definition>
</Term>

<Term termId="MAX">
    <Name xml:lang="en">MAX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing 3ds Max Scene File
    </Definition>
</Term>

<Term termId="OBJ">
    <Name xml:lang="en">OBJ</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Wavefront 3D Object File
    </Definition>
</Term>

<Term termId="BMP">
    <Name xml:lang="en">BMP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Bitmap Image File
    </Definition>
</Term>

<Term termId="DDS">
    <Name xml:lang="en">DDS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing DirectDraw Surface
    </Definition>
</Term>

<Term termId="GIF">
    <Name xml:lang="en">GIF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Graphical Interchange Format File
    </Definition>
</Term>

<Term termId="JPG">
    <Name xml:lang="en">JPG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing JPEG Image
    </Definition>
</Term>

<Term termId="PNG">
    <Name xml:lang="en">PNG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Portable Network Graphic
    </Definition>
</Term>

<Term termId="PSD">
    <Name xml:lang="en">PSD</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Adobe Photoshop Document
```

```
    </Definition>
</Term>

<Term termId="PSPIMAGE">
    <Name xml:lang="en">PSPIMAGE</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing PaintShop Pro Image
    </Definition>
</Term>

<Term termId="TGA">
    <Name xml:lang="en">TGA</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Targa Graphic
    </Definition>
</Term>

<Term termId="THM">
    <Name xml:lang="en">THM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Thumbnail Image File
    </Definition>
</Term>

<Term termId="TIF">
    <Name xml:lang="en">TIF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Tagged Image File
    </Definition>
</Term>

<Term termId="TIFF">
    <Name xml:lang="en">TIFF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Tagged Image File Format
    </Definition>
</Term>

<Term termId="YUV">
    <Name xml:lang="en">YUV</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing YUV Encoded Image File
    </Definition>
</Term>

<Term termId="AI">
    <Name xml:lang="en">AI</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Adobe Illustrator File
    </Definition>
</Term>

<Term termId="EPS">
    <Name xml:lang="en">EPS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Encapsulated PostScript File
    </Definition>
</Term>
```

```
<Term termId="PS">
    <Name xml:lang="en">PS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing PostScript File
    </Definition>
</Term>

<Term termId="SVG">
    <Name xml:lang="en">SVG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Scalable Vector Graphics File
    </Definition>
</Term>

<Term termId="INDD">
    <Name xml:lang="en">INDD</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Adobe InDesign Document
    </Definition>
</Term>

<Term termId="PCT">
    <Name xml:lang="en">PCT</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Picture File
    </Definition>
</Term>

<Term termId="PDF">
    <Name xml:lang="en">PDF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Portable Document Format File
    </Definition>
</Term>

<Term termId="XLR">
    <Name xml:lang="en">XLR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Works Spreadsheet
    </Definition>
</Term>

<Term termId="XLS">
    <Name xml:lang="en">XLS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Excel Spreadsheet
    </Definition>
</Term>

<Term termId="XLSX">
    <Name xml:lang="en">XLSX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Microsoft Excel Open XML Spreadsheet
    </Definition>
</Term>

<Term termId="ACCDB">
    <Name xml:lang="en">ACCDB</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Access 2007 Database File
```

```
    </Definition>
</Term>

<Term termId="DB">
    <Name xml:lang="en">DB</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Database File
    </Definition>
</Term>

<Term termId="DBF">
    <Name xml:lang="en">DBF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Database File
    </Definition>
</Term>

<Term termId="MDB">
    <Name xml:lang="en">MDB</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Microsoft Access Database
    </Definition>
</Term>

<Term termId="PDB">
    <Name xml:lang="en">PDB</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Program Database
    </Definition>
</Term>

<Term termId="SQL">
    <Name xml:lang="en">SQL</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Structured Query Language Data File
    </Definition>
</Term>

<Term termId="APK">
    <Name xml:lang="en">APK</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Android Package File
    </Definition>
</Term>

<Term termId="APP">
    <Name xml:lang="en">APP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Mac OS X Application
    </Definition>
</Term>

<Term termId="BAT">
    <Name xml:lang="en">BAT</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing DOS Batch File
    </Definition>
</Term>
```

```
<Term termId="CGI">
    <Name xml:lang="en">CGI</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Common Gateway Interface Script
    </Definition>
</Term>

<Term termId="COM">
    <Name xml:lang="en">COM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing DOS Command File
    </Definition>
</Term>

<Term termId="EXE">
    <Name xml:lang="en">EXE</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Windows Executable File
    </Definition>
</Term>

<Term termId="GADGET">
    <Name xml:lang="en">GADGET</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Windows Gadget
    </Definition>
</Term>

<Term termId="JAR">
    <Name xml:lang="en">JAR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Java Archive File
    </Definition>
</Term>

<Term termId="WSF">
    <Name xml:lang="en">WSF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Windows Script File
    </Definition>
</Term>

<Term termId="B">
    <Name xml:lang="en">B</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Grand Theft Auto 3 Saved Game File
    </Definition>
</Term>

<Term termId="DEM">
    <Name xml:lang="en">DEM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Video Game Demo File
    </Definition>
</Term>

<Term termId="GAM">
    <Name xml:lang="en">GAM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Saved Game File
```

```
    </Definition>
</Term>

<Term termId="NES">
    <Name xml:lang="en">NES</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Nintendo (NES) ROM File
    </Definition>
</Term>

<Term termId="ROM">
    <Name xml:lang="en">ROM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing N64 Game ROM File
    </Definition>
</Term>

<Term termId="SAV">
    <Name xml:lang="en">SAV</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Saved Game
    </Definition>
</Term>

<Term termId="DWG">
    <Name xml:lang="en">DWG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing AutoCAD Drawing Database File
    </Definition>
</Term>

<Term termId="DXF">
    <Name xml:lang="en">DXF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Drawing Exchange Format File
    </Definition>
</Term>

<Term termId="GPX">
    <Name xml:lang="en">GPX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing GPS Exchange File
    </Definition>
</Term>

<Term termId="KML">
    <Name xml:lang="en">KML</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Keyhole Markup Language File
    </Definition>
</Term>

<Term termId="KMZ">
    <Name xml:lang="en">KMZ</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Google Earth Placemark File
    </Definition>
</Term>
```

```
<Term termId="ASP">
    <Name xml:lang="en">ASP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Active Server Page
    </Definition>
</Term>

<Term termId="ASPX">
    <Name xml:lang="en">ASPX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Active Server Page Extended File
    </Definition>
</Term>

<Term termId="CER">
    <Name xml:lang="en">CER</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Internet Security Certificate
    </Definition>
</Term>

<Term termId="CFM">
    <Name xml:lang="en">CFM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing ColdFusion Markup File
    </Definition>
</Term>

<Term termId="CSR">
    <Name xml:lang="en">CSR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Certificate Signing Request File
    </Definition>
</Term>

<Term termId="CSS">
    <Name xml:lang="en">CSS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Cascading Style Sheet
    </Definition>
</Term>

<Term termId="DCR">
    <Name xml:lang="en">DCR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Shockwave Media File
    </Definition>
</Term>

<Term termId="HTM">
    <Name xml:lang="en">HTM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Hypertext Markup Language File
    </Definition>
</Term>

<Term termId="HTML">
    <Name xml:lang="en">HTML</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Hypertext Markup Language File
```

```
    </Definition>
</Term>

<Term termId="JS">
    <Name xml:lang="en">JS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing JavaScript File
    </Definition>
</Term>

<Term termId="JSP">
    <Name xml:lang="en">JSP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Java Server Page
    </Definition>
</Term>

<Term termId="PHP">
    <Name xml:lang="en">PHP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing PHP Source Code File
    </Definition>
</Term>

<Term termId="RSS">
    <Name xml:lang="en">RSS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Rich Site Summary
    </Definition>
</Term>

<Term termId="XHTML">
    <Name xml:lang="en">XHTML</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Extensible Hypertext Markup Language
File
    </Definition>
</Term>

<Term termId="CRX">
    <Name xml:lang="en">CRX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Chrome Extension
    </Definition>
</Term>

<Term termId="PLUGIN">
    <Name xml:lang="en">PLUGIN</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Mac OS X Plugin
    </Definition>
</Term>

<Term termId="FNT">
    <Name xml:lang="en">FNT</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Windows Font File
    </Definition>
</Term>
```

```
<Term termId="FON">
    <Name xml:lang="en">FON</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Generic Font File
    </Definition>
</Term>

<Term termId="OTF">
    <Name xml:lang="en">OTF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing OpenType Font
    </Definition>
</Term>

<Term termId="KMZ">
    <Name xml:lang="en">KMZ</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Google Earth Placemark File
    </Definition>
</Term>

<Term termId="ASP">
    <Name xml:lang="en">ASP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Active Server Page
    </Definition>
</Term>

<Term termId="ASPX">
    <Name xml:lang="en">ASPX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Active Server Page Extended File
    </Definition>
</Term>

<Term termId="CER">
    <Name xml:lang="en">CER</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Internet Security Certificate
    </Definition>
</Term>

<Term termId="CFM">
    <Name xml:lang="en">CFM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing ColdFusion Markup File
    </Definition>
</Term>

<Term termId="CSR">
    <Name xml:lang="en">CSR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Certificate Signing Request File
    </Definition>
</Term>

<Term termId="CSS">
    <Name xml:lang="en">CSS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Cascading Style Sheet
```

```
    </Definition>
</Term>

<Term termId="DCR">
    <Name xml:lang="en">DCR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Shockwave Media File
    </Definition>
</Term>

<Term termId="HTM">
    <Name xml:lang="en">HTM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Hypertext Markup Language File
    </Definition>
</Term>

<Term termId="HTML">
    <Name xml:lang="en">HTML</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Hypertext Markup Language File
    </Definition>
</Term>

<Term termId="JS">
    <Name xml:lang="en">JS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing JavaScript File
    </Definition>
</Term>

<Term termId="JSP">
    <Name xml:lang="en">JSP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Java Server Page
    </Definition>
</Term>

<Term termId="PHP">
    <Name xml:lang="en">PHP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing PHP Source Code File
    </Definition>
</Term>

<Term termId="RSS">
    <Name xml:lang="en">RSS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Rich Site Summary
    </Definition>
</Term>

<Term termId="XHTML">
    <Name xml:lang="en">XHTML</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Extensible Hypertext Markup Language
File
    </Definition>
</Term>
```

```
<Term termId="CRX">
    <Name xml:lang="en">CRX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Chrome Extension
    </Definition>
</Term>

<Term termId="PLUGIN">
    <Name xml:lang="en">PLUGIN</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Mac OS X Plugin
    </Definition>
</Term>

<Term termId="FNT">
    <Name xml:lang="en">FNT</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Windows Font File
    </Definition>
</Term>

<Term termId="FON">
    <Name xml:lang="en">FON</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Generic Font File
    </Definition>
</Term>

<Term termId="OTF">
    <Name xml:lang="en">OTF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing OpenType Font
    </Definition>
</Term>

<Term termId="GZ">
    <Name xml:lang="en">GZ</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Gnu Zipped Archive
    </Definition>
</Term>

<Term termId="PKG">
    <Name xml:lang="en">PKG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Mac OS X Installer Package
    </Definition>
</Term>

<Term termId="RAR">
    <Name xml:lang="en">RAR</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing WinRAR Compressed Archive
    </Definition>
</Term>

<Term termId="RPM">
    <Name xml:lang="en">RPM</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Red Hat Package Manager File
```

```
    </Definition>
</Term>

<Term termId="SITX">
    <Name xml:lang="en">SITX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing StuffIt X Archive
    </Definition>
</Term>

<Term termId="TARGZ">
    <Name xml:lang="en">TARGZ</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Compressed Tarball File
    </Definition>
</Term>

<Term termId="ZIP">
    <Name xml:lang="en">ZIP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Zipped File
    </Definition>
</Term>

<Term termId="ZIPX">
    <Name xml:lang="en">ZIPX</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Extended Zip File
    </Definition>
</Term>

<Term termId="BIN">
    <Name xml:lang="en">BIN</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Binary Disc Image
    </Definition>
</Term>

<Term termId="CUE">
    <Name xml:lang="en">CUE</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Cue Sheet File
    </Definition>
</Term>

<Term termId="DMG">
    <Name xml:lang="en">DMG</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Mac OS X Disk Image
    </Definition>
</Term>

<Term termId="ISO">
    <Name xml:lang="en">ISO</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Disc Image File
    </Definition>
</Term>
```

```
<Term termId="MDF">
    <Name xml:lang="en">MDF</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Media Disc Image File
    </Definition>
</Term>

<Term termId="TOAST">
    <Name xml:lang="en">TOAST</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Toast Disc Image
    </Definition>
</Term>

<Term termId="VCD">
    <Name xml:lang="en">VCD</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Virtual CD
    </Definition>
</Term>

<Term termId="C">
    <Name xml:lang="en">C</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing C/C++ Source Code File
    </Definition>
</Term>

<Term termId="CLASS">
    <Name xml:lang="en">CLASS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Java Class File
    </Definition>
</Term>

<Term termId="CPP">
    <Name xml:lang="en">CPP</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing C++ Source Code File
    </Definition>
</Term>

<Term termId="CS">
    <Name xml:lang="en">CS</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing C# Source Code File
    </Definition>
</Term>

<Term termId="DTD">
    <Name xml:lang="en">DTD</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Document Type Definition File
    </Definition>
</Term>

<Term termId="FLA">
    <Name xml:lang="en">FLA</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Adobe Animate Animation
```

```
    </Definition>
</Term>

<Term termId="H">
    <Name xml:lang="en">H</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing C/C++/Objective-C Header File
    </Definition>
</Term>

<Term termId="JAVA">
    <Name xml:lang="en">JAVA</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Java Source Code File
    </Definition>
</Term>

<Term termId="LUA">
    <Name xml:lang="en">LUA</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Lua Source File
    </Definition>
</Term>

<Term termId="M">
    <Name xml:lang="en">M</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Objective-C Implementation File
    </Definition>
</Term>

<Term termId="PL">
    <Name xml:lang="en">PL</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Perl Script
    </Definition>
</Term>

<Term termId="PY">
    <Name xml:lang="en">PY</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Python Script
    </Definition>
</Term>

<Term termId="SH">
    <Name xml:lang="en">SH</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Bash Shell Script
    </Definition>
</Term>

<Term termId="SLN">
    <Name xml:lang="en">SLN</Name>
    <Definition xml:lang="en">
       An MStorage is capable of storing Visual Studio Solution File
    </Definition>
</Term>
```