International Standard

ISO/IEC 23093-3

Information technology — Internet of media things —

Part 3: Media data formats and application programming interface (API)

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

Third edition
2026-07

Reference number
ISO/IEC 23093-3:2026(en)

COPYRIGHT PROTECTED DOCUMENT

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).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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 third edition cancels and replaces the second edition (ISO/IEC 23093-3:2022), which has been technically revised.

The main changes are as follows:

— Addition of new APIs and data formats;

— Addition of MController base types;

— Removal of MAnalyser APIs and individual data formats.

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 (MThings).

The APIs for MThings facilitate the discovery of other MThings in the network, as well as the connection and efficient exchange of data between MThings. The APIs also support transaction tokens to access valuable functionalities, resources, and data from MThings.

MThing-related information comprises characteristics and discovery data, mission descriptions from system designers and end-users, raw and processed sensed data, and actuation information. The ISO/IEC 23093 series specifies input and output data formats for media sensors, actuators, storages, and analysers. In addition, media analysers can process sensed data from media sensors to produce analysed data, which can be cascaded to other media analysers to extract semantic information. Multiple MThings can be gathered and operated autonomously using mission descriptions given by system designers and end-users.

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

Information technology — Internet of media things —

Part 3: Media data formats and application programming interface (API)

1 Scope

This document specifies the syntax and semantics of description schemes for representing data exchanged by media things (e.g. media sensors, media actuators, and media storages). Moreover, it specifies the APIs to exchange these data between media things. The following interfaces are under the scope of this document:

— APIs for accessing media sensors, actuators, storage, managers, controllers, and aggregators;

— structured data formats (XML) representing media thing's base data types, including sensors, actuators, storage, managers, aggregators, controllers, and their elements;

— structured data formats (XML) representing the media sensor's output data; and,

— structured data formats (XML) representing media actuator's commands.

This document does not specify how sensing and actuating are 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 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 23093-2:2025, *Information technology — Internet of media things — Part 2: Discovery and communication API*

ISO/IEC 23093-5:2025, *Information technology — Internet of media things — Part 5: IoMT autonomous collaboration*

ISO/IEC 23093-6:2026, *Information technology — Internet of media things — Part 6: IoMT Media data formats and API for distributed AI processing*

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 standardisation at the following addresses:

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

— IEC Electropedia: available at https://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

3.1.7
media thing controller
mThing controller
MController
MThing that can generate standard mission descriptions, control and monitor the progress of the mission of the participating MThings

3.1.8
mission
task that MThings carry out

3.2 Abbreviated terms

API	application programming interface
MACV	media actuator command vocabulary
MAOV	media analyser output vocabulary
MCOV	media thing controller output vocabulary
MSOV	media sensor output vocabulary
MTDL	media thing description language
OOP	object-oriented programming
SCDV	sensor capability description vocabulary
XML	extensible mark-up language
XSI	XML schema instance

4 Schema documents and prefixes

4.1 Schema documents

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

In some cases, though, particularly for Clauses 7 and 8, the syntax of data interfacing media things 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 available at: https://standards.iso.org/iso-iec/23093/-3/ed-3/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.

4.2 Use of prefixes

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

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

"`xml:`" and "`xmlns:`" are normative prefixes defined in Reference [1]. The prefix "`xml:`" is bound to "`https://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:`", "`scdv:`".

In particular, most informative examples in this document are provided as XML fragments without the typically required XML document declaration, and thus, they miss a correct namespace binding context declaration. The different prefixes are bound to the namespaces in these description fragments, 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
mtdl	urn:mpeg:mpeg-IoMT:2024:01-MTDL-NS
msov	urn:mpeg:mpeg-IoMT:2024:01-MSOV-NS
macv	urn:mpeg:mpeg-IoMT:2024:01-MACV-NS
maov	urn:mpeg:mpeg-IoMT:2024:01-MAOV-NS
mcov	Urn:mpeg:mpeg-IoMT:2024:01-MCOV-NS
xsi	https://www.w3.org/2001/XMLSchema-instance
xsd	https://www.w3.org/2001/XMLSchema

Unlike informative descriptions and 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 subclause 6.2.

5 APIs

5.1 General

This subclause specifies APIs and their descriptions for operating MThings and exchanging structured data between MThings.

Figure 1 shows an example of the "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 an MAnalyser. An 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, an MActuator should provide "SET" functions to control its functionalities. If no structured data is 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) returns 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 represented in either XML or binary format.

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 an MAnalyser can be delivered in either a simple string, such as a URL, or in the return type class "AnalysedDataType," which can be represented in XML or binary format.

Finally, the face verifier (AS2) invokes the `setLight()` function to actuate (i.e. generate the coloured light) the lighting device (AC1). Again, the actuation data feeding to an 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 to generate and exchange data or control MThings.

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

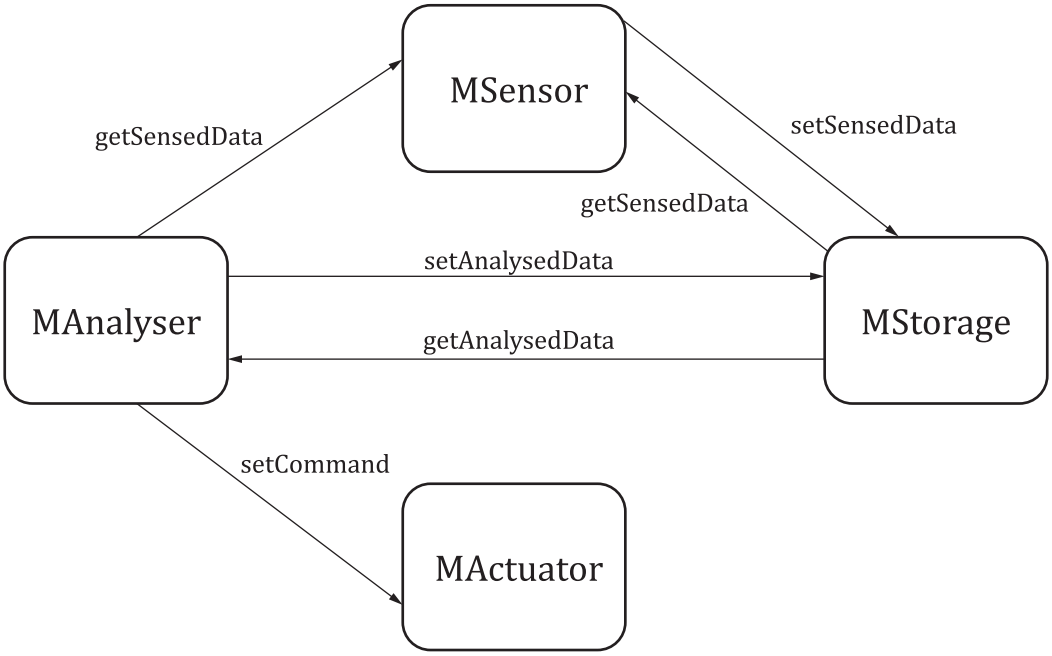

Figure 1 — Function invocation between MThings

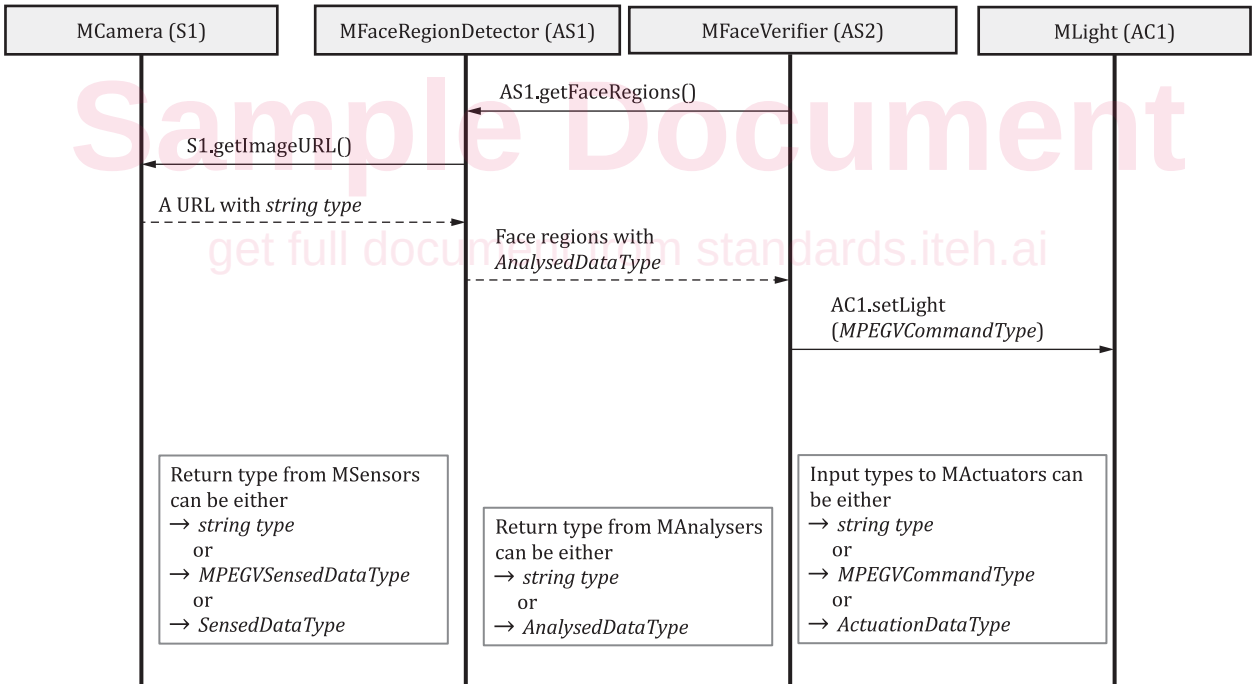

Figure 2 — Sequence diagram of function calls between MThings

5.2 APIs for IoMT sensors

5.2.1 General

This subclause defines the API classes of IoMT sensors.

5.2.2 MSensor class

5.2.2.1 General

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

5.2.2.2 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. OOP like 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. OOP like 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. OOP like 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.
CapabilityListType	getAvailableSensorCapabilityList()
	This function returns a class (i.e. OOP like 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.
CapabilityListType	getAppliedSensorCapabilityList()
	This function returns a class (i.e. OOP like 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.
SensedDataType	getCapturedTime()
	This function returns a captured time of sensed data.
SensedDataType	getCapturedTime(string tid)

Table 2 *(continued)*

	This function returns a captured time of sensed data. The `tid` is the transaction ID of payment for using this function.
string	getSubscriptionUrl()
	This function returns the subscription URL to receive events from this MSensor. This function will return "Not Supported" if it's not supported. *Ex) mqtt://mysensor:5050/mytopic*
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 subclause A.4.2. If the requested token is not listed in `TokenTypeCS`, the `tokenName` can be inserted manually using the conventional cryptocurrency name from the market. If the requested token is not supported, it returns –1. Ex) getCapturedTime_Cost(0, "BTC") or getCapturedTime_Cost(0, "00000001") Ex) getCapturedTime_Cost(1, "USD") or getCapturedTime_Cost(1, "10010100")

5.2.3 API for IoMT microphone

5.2.3.1 General

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

5.2.3.2 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()