

IEEE Standard for Specification of Sensor Interface for Cyber and Physical Worlds

IEEE Computer Society

Developed by the
Standards Activities Board (SAB) Standard Committee

IEEE Std 2888.1™-2023

STANDARDS

IEEE Standard for Specification of Sensor Interface for Cyber and Physical Worlds

Developed by the

Standard Activities Board (SAB) Standard Committee
of the
IEEE Computer Society

Approved 8 November 2023

IEEE SA Standards Board

Abstract: The vocabulary, data formats, and application programming interfaces (APIs) for acquiring information from sensors are defined in this standard, enabling communication between the cyber world and the physical world, including:

- Data formats for interfacing sensor data defining a root schema with individual sensor data
- Data formats for recognizing sensor capabilities defining a root schema with individual sensor capabilities
- Data formats for common data types
- Application programming interfaces to transmit a sensor and its capability data between cyber and physical worlds

Keywords: audio-visual sensors, biosensors, environmental sensors, IEEE 2888.1™, large space virtual reality (VR) training system sensors, location and position sensors, sensor capabilities, sensor data formats, sensor application programming interfaces (APIs)

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2024 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 8 January 2024. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 979-8-8557-0301-6 STD26605
Print: ISBN 979-8-8557-0302-3 STDPD26605

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA, and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and Standards Coordinating Committees are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#). An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#). For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#). Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the [IEEE SA Patent Policy](#).

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at <https://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Participants

At the time this draft Standard was completed, the Interfacing Cyber and Physical World Working Group had the following membership:

Kyoungro Yoon, *Chair*
Sang-Kyun Kim, *Vice Chair/Task Group Chair*
SangKwon Jeong, *Secretary*
Minhyuk Jeong, *Technical Editor*

Hyochul Bae
Dong Soo Choi
JeongHwoan Choi

Hoe Yong Jin
Shin Kim
Tai-Gil Kwon
Gookhwan Lee
Yegi Lee

Tae-Beom Lim
HyeonWoo Nam
Kwanghyun Ro
Changseok Yoon

The following members of the individual Standards Association balloting group voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Boon Chong Ang
Steven Bezner
Pin Chang
Diego Chiozzi
JeongHwoan Choi
Donald Dunn
Madhukar Gaganam
Andrew Min-gyu Han
Werner Hoelzl

Tyler Jaynes
SangKwon Jeong
Piotr Karocki
Mikhail Lagoda
Seung Wook Lee
Rajesh Murthy
HyeonWoo Nam
George Percivall
Venkatesha Prasad

R. K. Rannow
Eugene Stoudenmire
Keith Waters
Karl Weber
Changseok Yoon
Kyoungro Yoon
Yu Yuan
Janusz Zalewski
Zicheng Zhang

When the IEEE SA Standards Board approved this standard on 8 November 2023, it had the following membership:

David J. Law, *Chair*
Ted Burse, *Vice Chair*
Gary Hoffman, *Past Chair*
Konstantinos Karachalios, *Secretary*

Sara R. Biyabani
Doug Edwards
Ramy Ahmed Fathy
Guido R. Hiertz
Yousef Kimiagar
Joseph L. Koepfinger*
Thomas Koshy
John D. Kulick

Joseph S. Levy
Howard Li
Johnny Daozhuang Lin
Gui Lin
Xiaohui Liu
Kevin W. Lu
Daleep C. Mohla
Andrew Myles

Paul Nikolich
Annette D. Reilly
Robby Robson
Lei Wang
F. Keith Waters
Karl Weber
Philip B. Winston
Don Wright

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 2888.1-2023, IEEE Standard for Specification of Sensor Interface for Cyber and Physical Worlds.

This standard defines the vocabulary, data formats, and application programming interfaces (APIs) for acquiring information from sensors, enabling the definition of interfaces between the cyber world and the physical world.

Contents

1. Overview	10
1.1 Scope	10
1.2 Word usage	10
2. Normative references.....	11
3. Definitions, acronyms, and abbreviations	11
3.1 Definitions	11
3.2 Acronyms and abbreviations	11
4. Data formats for interfacing sensor data.....	12
4.1 Root schema structure.....	12
4.2 Data formats for individual sensor data	17
5. Data formats for recognizing sensor capabilities.....	61
5.1 Root schema structure.....	61
5.2 Data formats for individual sensor capabilities.....	65
6. Common data types	100
6.1 General	100
6.2 integer3DVector type.....	100
6.3 float3DVector type	101
6.4 float4DVector type	102
6.5 Unit type	102
6.6 Color type	105
6.7 Gas type	124
7. Application programming interface for smart sensors.....	125
7.1 General	125
7.2 Application programming interface for individual sensors	125

IEEE Standard for Specification of Sensor Interface for Cyber and Physical Worlds

1. Overview

1.1 Scope

This standard defines the vocabulary, data formats, and application programming interfaces (APIs) for acquiring information from sensors, enabling communication between the cyber world and the physical world, including:

- Data formats for interfacing sensor data defining a root schema with individual sensor data
- Data formats for recognizing sensor capabilities defining a root schema with individual sensor capabilities
- Data formats for common data types
- Application programming interfaces to transmit a sensor and its capability data between cyber and physical worlds

1.2 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).^{1, 2}

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

¹ The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.

² The use of *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

ISO/IEC 14496-1:2010/Amd.2:2014(E) Information technology—Coding of audio-visual objects—Part 1: Systems AMENDMENT 2: Support for raw audio-visual data.³

ISO/IEC 21778:2017(en) Information technology—The JSON data interchange syntax.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.⁴

sensor: Device by which user input or environmental information can be gathered.

sensor capability: Description of representing the characteristics of sensors in terms of the capacity of the given sensor, such as accuracy, or sensing range.

3.2 Acronyms and abbreviations

3D	three dimensions
4D	four dimensions
API	application programming interface
crs	coordinate reference system
GPS	global positioning system
HSV	hue saturation value
ID	identifier
IMU	inertial measurement unit
JSON	JavaScript object notation
PCM	pulse code modulation
pH	potential of hydrogen
RGB	red, green, blue
RGBW	red, green, blue, white
SI	Système International d'Unités or International System of Units
URI	uniform resource identifier
UV	ultraviolet
VR	virtual reality

³ ISO publications are available from the International Organization for Standardization (<https://www.iso.org/>) and the American National Standards Institute (<https://www.ansi.org/>). IEC publications are available from the International Electrotechnical Commission (<https://www.iec.ch>) and the American National Standards Institute (<https://www.ansi.org/>).

⁴ *IEEE Standards Dictionary Online* is available at: <http://dictionary.ieee.org>. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

4. Data formats for interfacing sensor data

The syntax and semantics of data formats for interfacing sensor data shall be indicated as shown in this clause. The syntax of sensor data shall be defined according to ISO/IEC 21778:2017(en).⁵

4.1 Root schema structure

4.1.1 General

This subclause specifies root schema that describes basic sensor data information.

4.1.2 Syntax

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Sensor data",
  "description": "Schema for sensor data",
  "type": "object",
  "properties": {
    "timeStamp": {
      "type": "string",
      "format": "date-time"
    },
    "aggregatorInfo": {
      "type": "object",
      "properties": {
        "sensorDataBaseAttributes": {
          "$ref": "#/definitions/sensorDataBaseAttributes"
        }
      }
    },
  },

  // This is where the properties of each type of sensor data are
  // declared by reference.
  // For Example,
  //   "microphoneSensorData": {
  //     "$ref": "#/definitions/microphoneSensorData"
  //   },
  //   "cameraSensorData": {
  //     "$ref": "#/definitions/cameraSensorData "
  //   },
}
```

⁵ Information on references can be found in Clause 2.

```
// ...
},
"additionalProperties": false,
"definitions": {
  "sensorDataBaseAttributes": {
    "additionalProperties": false,
    "type": "object",
    "properties": {
      "id": {"type": "string"},
      "sensorIdRef": {"type": "string"},
      "sensorList": {"type": "string"},
      "sensorGroupID": {"type": "string"},
      "activated": {"type": "boolean"},
      "priority": {"type": "integer", "minimum": 1},
      "relativeTime": {
        "type": "object",
        "properties": {
          "value": {"type": "integer"},
          "unit": {
            "$ref": "#/definitions/unitType"
          }
        }
      }
    }
  }
},

// This is where the properties of each type of sensor data are
defined.
// For Example,
//   "microphoneSensorData": {
//     ...
//   },
//   "cameraSensorData": {
//     ...
//   },
//   ...
// }
}
```

4.1.3 Semantics

The semantics of the root element:

Name	Definition
sensorDataBaseAttributes	It describes a group of base attributes for the sensor data.
timeStamp	It provides information about the time the sensor data was acquired.
aggregatorInfo	It describes the information of an aggregator that can have multiple sensors in it.
Id	It uniquely identifies individual sensor data.
sensorIdRef	It provides the sensor device unique ID that generated the data contained in this instance.
sensorList	It describes a multisensor structure organized into sensor groups so that each record contains a reference to the ID of the following sensor.
sensorGroupID	It identifies a group multisensor structure to which this specific sensor belongs.
activated	It describes whether the sensor is activated. A value of "true" means the sensor is active and "false" means the sensor is disabled.
priority	It describes the priority of sensor data with respect to other sensor data that share the same time when sensor data is adapted. A value of 1 represents the highest priority, and a larger value represents the lower priority. If there is more than one sensor data of the same priority, the order of processing can be determined by the adaptation engine itself.
relativeTime	It describes the relative time from timeStamp. If the relativeTime is later than timeStamp, it is described with a positive value, and if it is before timeStamp, it is described with a negative value.

4.1.4 Examples

There are three ways to describe sensor data. The first is to define a single data acquired from a single sensor. The second is to describe time series data received from a single sensor. The third is to describe a single data or time series data from an aggregator with multiple sensors.

In this example, a blood pressure sensor measured 121 mmHg of systolic blood pressure and 83 mmHg of diastolic blood pressure at 18:25:43 on 23/04/2020. The metadata ID is "BP000000017", and the priority is 1.

```
{
  "timeStamp": "2020-04-23T18:25:43.511Z",
  "bloodPressureSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "BP000000017",
      "sensorIdRef": "ieee2888sensors.org/BP-003",
      "activated": true,
      "priority": 1
    },
    "systolicBP": 121,
    "diastolicBP": 83,
    "MAP": 100
  }
}
```

In this example, a heart rate sensor measured 98 BPM, 101 BPM, and 106 BPM consecutively from 18:25:43.5 on 23/04/2020 every 0.5 s.

```
{
  "timeStamp": "2020-04-23T18:25:43.511Z",
  "heartRateSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "HR000000023",
      "sensorIdRef": "ieee2888sensors.org/HR-001",
      "activated": true,
      "relativeTime": {
        "value": 0,
        "unit": millisecond
      }
    },
    "value": 98
  }
  "heartRateSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "HR000000024",
      "sensorIdRef": "ieee2888sensors.org/HR-001",
      "activated": true,
      "relativeTime": {
        "value": 500,
        "unit": millisecond
      }
    },
    "value": 101
  }
  "heartRateSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "HR000000025",
      "sensorIdRef": "ieee2888sensors.org/HR-001",
      "activated": true,
      "relativeTime": {
        "value": 1000,
        "unit": millisecond
      }
    },
    "value": 106
  }
}
```

```
"value": 106
}
}
```

In this example, an aggregator containing a heart rate sensor and a blood pressure sensor acquired time series data from its sensors. The heart rate sensor measured 98 BPM of heart rate after 1 se from the time stamp, and the blood pressure sensor measured 121 mmHg of systolic blood pressure and 83 mmHg of diastolic blood pressure after 3 s of the time stamp.

```
{
  "timeStamp": "2020-04-23T18:25:43.511Z",
  "aggregatorInfo": {
    "sensedInfoBaseAttributes": {
      "id": "BIO202004231828430001",
      "sensorIdRef": "ieee2888sensors.org/BIO-001",
      "activated": true,
      "priority": 1
    }
  }
  "heartRateSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "HR000000023",
      "sensorIdRef": "ieee2888sensors.org/HR-001",
      "activated": true,
      "relativeTime": {
        "value": 1000,
        "unit": millisecond
      }
    },
    "value": 98
  }
  "bloodPressureSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "BP000000017",
      "sensorIdRef": "ieee2888sensors.org/BP-003",
      "activated": true,
      "relativeTime": {
        "value": 3000,
        "unit": millisecond
      }
    }
  }
}
```

```
    }  
  },  
  "systolicBP": 121,  
  "diastolicBP": 83,  
  "MAP": 100,  
  "relativeTime": 10  
}  
}  
}
```

4.2 Data formats for individual sensor data

4.2.1 General

This subclause specifies individual sensor data types.

4.2.2 Audio-visual sensors

4.2.2.1 Microphone sensor

4.2.2.1.1 General

This subclause specifies a sensor data type that describes audio information captured from a microphone sensor.

4.2.2.1.2 Syntax

```
"microphoneSensorData": {  
  "type": "object",  
  "properties": {  
    "sensorDataBaseAttributes": {  
      "$ref": "#/definitions/sensorDataBaseAttributes"  
    },  
    "microphoneOrientation": {  
      "$ref": "#/definitions/orientationSensorData"  
    },  
    "microphoneLocation": {  
      "$ref": "#/definitions/globalPositionSensorData "  
    },  
    "microphoneAltitude": {  
      "$ref": "#/definitions/altitudeSensorData"  
    },  
    "audioData": {
```

```
        "$ref": "#/definitions/rawAudioType"
    },
    },
    "additionalProperties": false
},
"rawAudioType": {
    "type": "object",
    "properties": {
        "sample_rate": {
            "$ref": "#/definitions/sampleRateType"
        },
        "byte_order": {
            "$ref": "#/definitions/byteOrderType"
        },
        "sign": {
            "$ref": "#/definitions/signType"
        },
        "resolution": {
            "$ref": "#/definitions/resolutionType"
        }
    },
    "additionalProperties": false
},
"sampleRateType": {
    "type": "number",
    "enum": [
        8.0,
        11.025,
        16.0,
        22.05,
        32.0,
        44.056,
        44.1,
        47.25,
        48.0,
        50.0,
        50.4,
        88.2,
```

```
    96.0,  
    176.4,  
    192.0,  
    352.8,  
    2822.4,  
    5644.8  
  ]  
,  
  "byteOrderType": {  
    "type": "string",  
    "enum": [  
      "littleEndian",  
      "bigEndian"  
    ]  
  },  
  "signType": {  
    "type": "string",  
    "enum": [  
      "signed",  
      "unsigned"  
    ]  
  },  
  "resolutionType": {  
    "type": "integer",  
    "enum": [  
      4,  
      8,  
      12,  
      16,  
      20,  
      24,  
      32,  
      48,  
      64  
    ]  
  },  
  },
```

4.2.2.1.3 Semantics

The semantics of the `microphoneSensorData`:

Name	Definition
<code>microphoneSensorData</code>	It provides a structure for describing data acquired by a microphone.
<code>microphoneLocation</code>	It describes the location of a microphone using the structure defined by <code>globalPositionSensorData</code> .
<code>microphoneOrientation</code>	It describes the orientation of a microphone using the structure defined by <code>orientationSensorData</code> .
<code>microphoneAltitude</code>	It describes the altitude of a microphone using the structure defined by <code>altitudeSensorData</code> .
<code>audioData</code>	It describes audio data referring to the <code>rawAudioType</code> .
<code>sampleRate</code>	It describes the number of samples of audio carried per second, measured in hertz (Hz) or kilohertz (kHz) (1 kHz being 1000 Hz). For example, 44,100 samples per second can be expressed as either 44 100 Hz or 44.1 kHz. Bandwidth is the difference between the highest and lowest frequencies carried in an audio stream.
<code>byteOrder</code>	It describes where the most significant byte is stored in the data. When more than one byte is used to represent a PCM sample, the byte order (big-endian vs. little-endian) shall be described with <code>littleEndian</code> and <code>bigEndian</code> .
<code>sign</code>	It describes whether the PCM sample is signed or unsigned. If the 8 bit sample is unsigned, the sample range is 0...255 with a center point of 128. If the 8 bit sample is signed, the sample range is -128...127 with a center point of 0. If a PCM type is signed, the sign encoding is 2's complement, signed and unsigned.
<code>quantizationBit</code>	It describes the number of bits assigned to represent each discrete amplitude sample. The possible values are 4, 8, 12, 16, 20, 24, 32, 48, and 64.
<code>signed</code>	It describes that the raw audio data coming from the microphone is stored as signed numbers.
<code>unsigned</code>	It describes that the raw audio data coming from the microphone is stored as unsigned numbers.
<code>bigEndian</code>	It describes that the audio data is stored in the big-endian format: the most significant byte of a word is in the smallest address, and the least significant byte is stored in the largest address.
<code>littleEndian</code>	It describes that the audio data is stored in the little-endian format: the least significant byte in the smallest address.

4.2.2.1.4 Examples

In this example, the orientation has (20, 10, 0), the location is (33.413, 38.623), and the altitude is 321.6 m. The sampling rate is 44.1 kHz, and the byte order follows the little-endian method. It is signed data with a resolution of 8 bits.

```
{
  "microphoneSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "MIC027"
    },
    "microphoneOrientation": {
      "orientation": [20, 10, 0]
    }
  }
}
```

```
    },  
    "microphoneLocation": {  
      "longitude": 33.413,  
      "latitude": 38.623  
    },  
    "microphoneAltitude": {  
      "altitude": 321.6,  
      "unit": "meter"  
    }  
  }  
  "audioData": {  
    "sample_rate": 44.1,  
    "byte_order": "littleEndian",  
    "sign": "signed",  
    "resolution": 8  
  }  
}  
}
```

4.2.2.2 Color camera sensor

4.2.2.2.1 General

This subclause specifies a sensor data type that describes the color of the captured objects in the environment, encoded as a sequence of pixel values.

4.2.2.2.2 Syntax

```
"colorCameraSensorData": {  
  "type": "object",  
  "properties": {  
    "sensorDataBaseAttributes": {  
      "$ref": "#/definitions/sensorDataBaseAttributes"  
    },  
    "cameraSensor": {  
      "$ref": "#/definitions/cameraSensorData"  
    },  
    "rawVideo": {  
      "$ref": "#/definitions/rawVideoType"  
    }  
  },  
  "additionalProperties": false  
}
```

```
},
"rawVideoType": {
  "type": "object",
  "properties": {
    "videoData16": {
      "type": "string"
    },
    "videoData64": {
      "type": "string"
    },
    "width": {
      "type": "integer"
    },
    "height": {
      "type": "integer"
    },
    "bit_depth": {
      "type": "integer"
    },
    "stride": {
      "type": "integer"
    },
    "coding4CC": {
      "type": "integer"
    },
    "fps": {
      "type": "integer"
    },
    "use_frame_packing": {
      "type": "boolean"
    },
    "frame_packing": {
      "type": "integer"
    }
  },
  "additionalProperties": false
},
"cameraSensorData": {
```

```
"type": "object",
"properties": {
  "cameraOrientation": {
    "$ref": "#/definitions/orientationSensorData"
  },
  "cameraLocation": {
    "$ref": "#/definitions/globalPositionSensorData"
  },
  "cameraAltitude": {
    "$ref": "#/definitions/altitudeSensorData"
  },
  "focalLength": {
    "type": "number"
  },
  "aperture": {
    "type": "number"
  },
  "shutterSpeed": {
    "type": "number"
  },
  "filter": {
    "type": "string",
    "enum": [
      "UV",
      "Polarizing",
      "NB",
      "Diffusion",
      "Star"
    ]
  },
  "additionalProperties": false
},
```

4.2.2.2.3 Semantics

The semantics of the `colorCameraSensorData`:

Name	Definition
<code>colorCameraSensorData</code>	It provides a structure for describing sensor data acquired by a color camera.
<code>rawVideo</code>	It describes the raw video data by the color camera.
<code>rawVideoType</code>	It provides a structure for describing raw video data acquired by a color camera.
<code>width</code>	It describes the width of the video in the number of pixels.
<code>height</code>	It describes the height of the video in the number of pixels.
<code>bitDepth</code>	It describes the number of bits for each channel sample from the set of permitted values as defined by <code>coding4CC</code> .
<code>stride</code>	It describes the size in bytes of one horizontal line.
<code>coding4CC</code>	It indicates a four-character code representing the parameters of the raw data as specified by ISO/IEC 14496-1:2010/Amd.2:2014(E).
<code>fps</code>	It describes frames per second of the video stream; if 0, then the frame rate is not known or variable.
<code>useFramePacking</code>	It indicates if a frame contains two or more views.
<code>framePacking</code>	It indicates frame Packing as Coding Independent Code Points.
<code>videoData16</code>	It holds binary video data encoded as a textual string in base-16 format.
<code>videoData64</code>	It holds binary video data encoded as a textual string in base-64 format.
<code>cameraSensorData</code>	It provides a structure for describing sensor data acquired by a camera.
<code>cameraLocation</code>	It describes the location of a camera using the structure defined by <code>globalPositionSensorData</code> .
<code>cameraOrientation</code>	It describes the orientation of a camera using the structure defined by <code>orientationSensorData</code> .
<code>cameraAltitude</code>	It describes the altitude of a camera using the structure defined by <code>altitudeSensorData</code> .
<code>focalLength</code>	It describes the distance between lens and image sensor when the subject is in focus, in terms of millimeters (mm).
<code>aperture</code>	It describes the diameter of the lens opening expressed as F-stop, e.g., F2.8. It may also be defined as f-number notation such as $f/2.8$.
<code>shutterSpeed</code>	It describes the time that the shutter remains open when taking a photograph in terms of seconds (s).
<code>filter</code>	It describes the kinds of camera filters.
<code>colorCameraData</code>	It provides a structure for describing sensor data acquired by a color camera.
<code>rawVideo</code>	It describes the raw video data by the color camera.

4.2.2.2.4 Examples

In this example, the camera orientation values are yaw, pitch, and roll values of 20, 10, and 0, respectively. The camera altitude is 321.6 m. The camera location has a longitude of 33.413 and a latitude of 38.623. The focal length is 55 mm, and the aperture is $f/2.8$. The shutter speed is 0.008 s, and the filter indicates that a UV filter is used.

The `videoData16` is base-16 encoded data starting with "0314BA3827CFF2938"...". The width and height of the image are 640 and 480 pixels, respectively. The `bit_depth` is 8, and the `stride` is 10. The `coding4CC` is 2, and the `fps` of the image is 30 frames/s. The `use_frame_packing` "s" "tr" "e," and the `frame_packing` is 10.

```
{  
  "colorCameraSensorData": {  
    "sensedInfoBaseAttributes": {},  
    "cameraSensor": {  
      "cameraOrientation": {  
        "orientation": [60, 30, 120]  
      },  
      "cameraLocation": {  
        "longitude": 33.413,  
        "latitude": 38.623  
      },  
      "cameraAltitude": {  
        "altitude": 321.6,  
        "unit": "meter"  
      }  
      "focalLength": 55,  
      "aperture": 2.8,  
      "shutterSpeed": 0.008,  
      "filter": "UV"  
    },  
    "rawVideo": {  
      "videoData16": "0314BA3827CFF2938...",  
      "width": 640,  
      "height": 480,  
      "bit_depth": 8,  
      "stride": 10,  
      "coding4CC": 2,  
      "fps": 30,  
      "use_frame_packing": true,  
      "frame_packing": 10  
    }  
  }  
}
```

4.2.3 Biosensors

4.2.3.1 Blood pressure sensor

4.2.3.1.1 General

This subclause specifies a sensor data type that describes blood pressure.

4.2.3.1.2 Syntax

```
"bloodPressureSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "systolicBP": {
      "type": "number"
    },
    "diastolicBP": {
      "type": "number"
    },
    "MAP": {
      "type": "number"
    },
    "unit": {
      "$ref": "#/definitions/unitType"
    }
  },
  "additionalProperties": false
},
```

4.2.3.1.3 Semantics

The semantics of the `bloodPressureSensorData`:

Name	Definition
<code>bloodPressureSensorData</code>	Tool for describing sensed data from a blood pressure sensor.
<code>systolicBP</code>	It describes the sensed value of the systolic blood pressure with the millimeters of mercury (mmHg).
<code>diastolicBP</code>	It describes the sensed value of the diastolic blood pressure with the millimeters of mercury (mmHg).
<code>MAP</code>	It describes the sensed value of the mean arterial pressure with the millimeters of mercury (mmHg).

4.2.3.1.4 Examples

In this example, the systolic blood pressure measured by the blood pressure sensor is 121 mmHg, the diastolic blood pressure is 83 mmHg, and the mean arterial pressure is 100 mmHg.

```
{
  "bloodPressureSensorData": {
    "sensedInfoBaseAttributes": {},
    "systolicBP": 121,
    "diastolicBP": 83,
    "MAP": 100
  }
}
```

4.2.3.2 Heart rate sensor

4.2.3.2.1 General

This subclause specifies a sensor data type that describes a heart rate.

4.2.3.2.2 Syntax

```
"heartRateSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "value": {
      "type": "number"
    },
    "unit": {
      "$ref": "#/definitions/unitType"
    }
  },
  "required": {
    "value"
  },
  "additionalProperties": false
},
```

4.2.3.2.3 Semantics

The semantics of the `heartRateSensorData`:

Name	Definition
<code>heartRateSensorData</code>	Tool for describing sensed data from a heart rate sensor.
<code>value</code>	It describes the sensed value of the heart rate with the beats per minute (BPM).

4.2.3.2.4 Examples

In this example, the heartbeat value measured by the heart rate sensor is 87 BPM.

```
{
  "heartRateSensorData": {
    "sensedInfoBaseAttributes": {
      "id": "HRS-070"
    },
    "value": 87
  }
}
```

4.2.4 Environment-related sensors

4.2.4.1 Ambient light sensor

4.2.4.1.1 General

This subclause specifies a sensor data type that describes light intensity and color.

4.2.4.1.2 Syntax

```
"ambientLightSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "value": {
      "$ref": "#/definitions/colorWType"
    },
    "unit": {
      "$ref": "#/definitions/unitType"
    },
    "color": {
```

```

    "%ref": "#/definitions/colorType"
  },
  "colorValue": {
    "%ref": "#/definitions/colorValueType"
  },
  "model": {
    "$ref": "#/definitions/colorSpaceType"
  }
},
"additionalProperties": false
},
"colorWType": {
  "type": "array",
  "items": [
    {
      "type": "number"
    },
    {
      "type": "string",
      "pattern": "#[0-9A-Fa-f]{2}"
    }
  ]
},
"colorValueType": {
  "type": "array",
  "items": [
    {
      "type": "number",
    }
  ],
  "minItems": 3,
  "maxItems": 3
},
"colorSpaceType": {
  "type": "string",
  "enum": [
    "XYZ",
    "Yxy",

```

```
"Lab",  
"Lch",  
"LUV",  
"HunterLab"  
]  
},
```

4.2.4.1.3 Semantics

The semantics of the `ambientLightSensorData`:

Name	Definition
<code>ambientLightSensorData</code>	It provides a structure for describing sensor data acquired by an ambient light sensor.
<code>value</code>	It describes the intensity of the light with the unit defined in the <code>unit</code> attribute. This attribute can represent "White" when the light sensor senses "RGBW".
<code>unit</code>	It specifies the unit of the value as a reference to a term that shall be using the <code>unitType</code> defined in 6.5.
<code>color</code>	It describes the list of colors the light sensor can sense as a reference to a term using the <code>colorType</code> .
<code>colorWType</code>	It provides a structure for describing a color in 1 byte (256 level) value of White.
<code>colorValue</code>	It describes the values of a color sensor with color space models.
<code>model</code>	It describes the color model of the values from a color sensor using the <code>colorSpaceType</code> .
<code>colorValueType</code>	It describes three values from a color sensor. The color space model determines the meaning of the three values. EXAMPLE: The color model CIEXYZ would have three values of X, Y, and Z in order.
<code>colorSpaceType</code>	The color space models utilized by a color sensor are Yxy, CIEXYZ, CIELAB, CIELCH, CIELUV, and Hunter Lab.

4.2.4.1.4 Examples

The light sensor detected 200 lux of light. Colors represent blue with 0.14, 0.8, and 0.78 in the CIE XYZ color space.

```
{  
  "ambientLightSensorData": {  
    "sensedInfoBaseAttributes": {},  
    "value": [200],  
    "unit": "lux",  
    "colorValue": [0.14, 0.8, 0.78],  
    "model": "XYZ"  
  }  
}
```

4.2.4.2 Ambient noise sensor

4.2.4.2.1 General

This subclause specifies a sensor data type that describes ambient noise and its duration.

4.2.4.2.2 Syntax

```
"ambientNoiseSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "lifespan": {
      "type": "number"
    },
    "value": {
      "type": "number"
    }
  },
  "additionalProperties": false
},
```

4.2.4.2.3 Semantics

The semantics of the `ambientNoiseSensorData`:

Name	Definition
<code>ambientNoiseSensorData</code>	It provides a structure for describing sensor data acquired by an ambient noise sensor.
<code>lifespan</code>	It describes the duration taken to measure the information based on the <code>timeStamp</code> . The unit of <code>lifespan</code> is the internal clock count.
<code>value</code>	It describes the sensed value of the ambient noise with the decibel (dB).

4.2.4.2.4 Examples

The ambient noise intensity is measured every 100 ticks of the clock in the ambient noise sensor. The ambient noise measured is 10 dB.

```
{
  "ambientNoiseSensorData": {
    "sensedInfoBaseAttributes": {},
    "lifespan": 100,
    "value": 10
  }
}
```

```
}  
}
```

4.2.4.3 Pressure sensor data

4.2.4.3.1 General

This subclause specifies a sensor data type that describes a pressure.

4.2.4.3.2 Syntax

```
"pressureSensorData": {  
  "type": "object",  
  "properties": {  
    "sensorDataBaseAttributes": {  
      "$ref": "#/definitions/sensorDataBaseAttributes"  
    },  
    "value": {  
      "type": "number"  
    },  
    "unit": {  
      "$ref": "#/definitions/unitType"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.4.3.3 Semantics

The semantics of the `pressureSensorData`:

Name	Definition
<code>pressureSensorData</code>	It provides a structure for describing sensor data acquired by a pressure sensor.
<code>value</code>	It describes the sensed pressure value by the pressure with the unit defined in the <code>unit</code> attribute.
<code>unit</code>	It specifies the unit of the value as a reference to a term provided by <code>unitType</code> .

4.2.4.3.4 Examples

In this example, the measured pressure value was 100 kPa.

```
{  
  "pressureSensorData": {  
    "sensedInfoBaseAttributes": {},
```

```
"value": 100,  
  "unit": "kiloPascal"  
}  
}
```

4.2.4.4 Temperature sensor

4.2.4.4.1 General

This subclause specifies a sensor data type that describes a temperature.

4.2.4.4.2 Syntax

```
"temperatureSensorData": {  
  "type": "object",  
  "properties": {  
    "sensorDataBaseAttributes": {  
      "$ref": "#/definitions/sensorDataBaseAttributes"  
    },  
    "value": {  
      "type": "number"  
    },  
    "unit": {  
      "$ref": "#/definitions/unitType"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.4.4.3 Semantics

The semantics of the `temperatureSensorData`:

Name	Definition
<code>temperatureSensorData</code>	It provides a structure for describing sensor data acquired by a temperature sensor.
<code>value</code>	It describes the temperature with the unit defined in the unit attribute.
<code>unit</code>	It specifies the unit of the temperature as a reference to a term that shall be using the <code>unitType</code> (Celsius or Fahrenheit).

4.2.4.4.4 Examples

The measured temperature is 36.5 °C.

```
{
  "temperatureSensorData": {
    "sensedInfoBaseAttributes": {},
    "value": 36.5,
    "unit": "celsius"
  }
}
```

4.2.4.5 Humidity sensor

4.2.4.5.1 General

This subclause specifies a sensor data type that describes humidity.

4.2.4.5.2 Syntax

```
"humiditySensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "value": {
      "type": "number"
    }
  },
  "additionalProperties": false
},
```

4.2.4.5.3 Semantics

The semantics of the `humiditySensorData`:

Name	Definition
<code>humiditySensorData</code>	It provides a structure for describing sensor data acquired by a humidity sensor.
<code>value</code>	It describes the value sensed by the humidity sensor with a percentage (%).

4.2.4.5.4 Examples

The humidity measured was 60%.

```
{
  "humiditySensorData": {
    "sensedInfoBaseAttributes": {},
    "value": 60
  }
}
```

4.2.4.6 Wind sensor

4.2.4.6.1 General

This subclause specifies a sensor data type known as an “anemometer,” which measures wind velocity at a specific position.

4.2.4.6.2 Syntax

```
"windSensorData": {
  "$ref": "#/definitions/velocitySensorData"
},
"velocitySensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "velocity": {
      "$ref": "#/definitions/float3DVectorType"
    },
    "unit": {
      "$ref": "#/definitions/unitType"
    }
  },
  "additionalProperties": false
},
```

4.2.4.6.3 Semantics

The semantics of the `windSensorData`:

Name	Definition
<code>windSensorData</code>	It provides a structure for describing sensor data acquired by a wind sensor.
<code>velocity</code>	It describes the speed and direction of a wind flow with the unit defined in the unit attribute. Values represent the wind speeds from the x-, y-, and z-directions, respectively.

Name	Definition
unit	It specifies the unit of the sensed value as a reference to a term that shall be using the <code>unitType</code> .

4.2.4.6.4 Examples

In this example, the velocities of wind measured were 1 m/s from the *x*-direction, 0.7 m/s from the *y*-direction, and 3.5 m/s from the *z*-direction.

```
{
  "windSensorData": {
    "sensedInfoBaseAttributes": {},
    "velocity": [1, 0.7, 3.5],
    "unit": "meterspersec"
  }
}
```

4.2.4.7 Gas sensor

4.2.4.7.1 General

This subclause specifies a sensor data type, describing a gas type and its concentration degree.

4.2.4.7.2 Syntax

```
"gasSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "gasType": {
      "$ref": "#/definitions/gasType"
    },
    "value": {
      "type": "number"
    },
    "unit": {
      "$ref": "#/definitions/unitType"
    },
    "additionalProperties": false
  },
}
```

4.2.4.7.3 Semantics

The semantics of the `gasSensorData`:

Name	Definition
<code>gasSensorData</code>	It provides a structure for describing sensor data acquired by a gas sensor.
<code>gasType</code>	It specifies the type of gas as a reference to a term provided by <code>gasType</code> .
<code>value</code>	It describes the sensed gas concentration value by the gas sensor with the unit defined in the unit attribute.
<code>unit</code>	It specifies the unit of the sensed value as a reference to a term provided by <code>unitType</code> .

4.2.4.7.4 Examples

In this example, the gas sensor measured 100 ppm of methane.

```
{
  "gasSensorData": {
    "sensedInfoBaseAttributes": {},
    "gasType": "methane",
    "value": 100,
    "unit": "partspermillion"
  }
}
```

4.2.4.8 Dust sensor

4.2.4.8.1 General

This subclause specifies a sensor data type that describes a dust concentration value without identifying the type of dust.

4.2.4.8.2 Syntax

```
"dustSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "value": {
      "type": "number"
    },
    "unit": {
```

```
"$ref": "#/definitions/unitType"
    }
  },
  "additionalProperties": false
},
```

4.2.4.8.3 Semantics

The semantics of the `dustSensorData`:

Name	Definition
<code>dustSensorData</code>	It provides a structure for describing sensor data acquired by a dust sensor.
<code>value</code>	It describes the dust concentration value by the dust check with the unit defined in the unit attribute.
<code>unit</code>	It specifies the unit of the value as a reference to a term provided by <code>unitType</code> .

4.2.4.8.4 Examples

The dust concentration value measured was 100 µg/m³.

```
{
  "dustSensorData": {
    "sensedInfoBaseAttributes": {},
    "value": 100,
    "unit": "microgpcm"
  }
}
```

4.2.4.9 Precipitation sensor

4.2.4.9.1 General

This subclause specifies a sensor data type that describes the amount of rainfall.

4.2.4.9.2 Syntax

```
"precipitationSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
    },
    "value": {
```

```
    "type": "number"
  },
  "unit": {
    "$ref": "#/definitions/unitType"
  }
},
"additionalProperties": false
},
```

4.2.4.9.3 Semantics

The semantics of the precipitationSensorData:

Name	Definition
precipitationSensorData	It provides a structure for describing sensor data acquired by a precipitation sensor.
value	It represents the measured value of the amount of rain.
unit	It specifies the unit of the value as a reference to a term provided by unitType, default as millimeters/hour (mm/h).

4.2.4.9.4 Examples

The measured rainfall is 100 mm/h.

```
{
  "precipitationSensorData": {
    "sensedInfoBaseAttributes": {},
    "value": 100,
    "unit": "millimetersperhour"
  }
}
```

4.2.4.10 Insolation sensor

4.2.4.10.1 General

This subclause specifies a sensor data type that describes insolation.

4.2.4.10.2 Syntax

```
"insolationSensorData": {
  "type": "object",
  "properties": {
    "sensorDataBaseAttributes": {
      "$ref": "#/definitions/sensorDataBaseAttributes"
```

```
    },  
    "value": {  
      "type": "number"  
    },  
    "unit": {  
      "$ref": "#/definitions/unitType"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.4.10.3 Semantics

The semantics of the `insolationSensorData`:

Name	Definition
<code>insolationSensorData</code>	It provides a structure for describing sensor data acquired by an insolation sensor.
<code>value</code>	It represents the measured value of the amount of insolation.
<code>unit</code>	It specifies the unit of the value as a reference to a term provided by <code>unitType</code> , default as Watts per square meter (W/m ²).

4.2.4.10.4 Examples

The measured insolation is 100 W/m².

```
{  
  "insolationSensorData": {  
    "sensedInfoBaseAttributes": {},  
    "value": 100,  
    "unit": "wattspersquaremeter"  
  }  
}
```

4.2.4.11 Soil moisture sensor

4.2.4.11.1 General

This subclause specifies a sensor data type that describes soil moisture.

4.2.4.11.2 Syntax

```
"soilMoistureSensorData": {  
  "type": "object",  
  "properties": {
```

```
"sensorDataBaseAttributes": {  
  "$ref": "#/definitions/sensorDataBaseAttributes"  
},  
"value": {  
  "type": "number"  
},  
"unit": {  
  "$ref": "#/definitions/unitType"  
}  
},  
"additionalProperties": false  
},
```

4.2.4.11.3 Semantics

The semantics of the `soilMoistureSensorData`:

Name	Definition
<code>soilMoistureSensorData</code>	It provides a structure for describing sensor data acquired by a soil moisture sensor.
<code>value</code>	It represents the measured value of the soil's moisture content per unit volume.
<code>unit</code>	It specifies the unit of the value as a reference to a term provided by <code>unitType</code> , default as a percentage (%).

4.2.4.11.4 Examples

The measured soil moisture is 1%.

```
{  
  "soilMoistureSensorData": {  
    "sensedInfoBaseAttributes": {},  
    "value": 1,  
    "unit": "percentage"  
  }  
}
```

4.2.4.12 Soil tensiometer

4.2.4.12.1 General

This subclause specifies a sensor data type that describes moisture tension in the soil.

4.2.4.12.2 Syntax

```
"soilTensiometerSensorData": {  
  "type": "object",  
  "properties": {  
    "sensorDataBaseAttributes": {  
      "$ref": "#/definitions/sensorDataBaseAttributes"  
    },  
    "value": {  
      "type": "number"  
    },  
    "unit": {  
      "$ref": "#/definitions/unitType"  
    }  
  },  
  "additionalProperties": false  
},
```

4.2.4.12.3 Semantics

The semantics of the `soilTensiometerSensorData`:

Name	Definition
<code>soilTensiometerSensorData</code>	It provides a structure for describing sensor data acquired by a soil tensiometer.
<code>value</code>	It represents the value of moisture tension in the soil.
<code>unit</code>	It specifies the unit of the value as a reference to a term provided by <code>unitType</code> , default as kilopascal (kPa).

4.2.4.12.4 Examples

The measured soil tensiometer is 1 kPa.

```
{  
  "soilTensiometerData": {  
    "sensedInfoBaseAttributes": {},  
    "value": 1,  
    "unit": "kiloPascal"  
  }  
}
```